

¡CONTIENE EJEMPLOS PRÁCTICOS PARA PONER MANOS A LA OBRA!

MICRO CONTROLADORES

FUNCIONAMIENTO, PROGRAMACIÓN Y APLICACIONES PRÁCTICAS

ADEMÁS

- LÓGICA DIGITAL
- MEMORIAS: TECNOLOGÍAS Y UTILIZACIÓN
- COMPUERTAS LÓGICAS, REGISTROS DE DESPLAZAMIENTO
- MICROCONTROLADORES: PIC16 EN ASSEMBLER Y PIC18 EN C

COLECCIÓN USERS ELECTRÓNICA


MICRO
CONTROLADORES
FUNCIONAMIENTO, PROGRAMACIÓN
Y APLICACIONES PRÁCTICAS
PIC16F887
-I/P
0937CQJ
MICROCHIP



TÍTULO: Microcontroladores

COLECCIÓN: desde Cero

FORMATO: 15 X 19 cm

PÁGINAS: 192

ISBN 978-987-1773-22-0

Microcontroladores / coordinado por Daniel Benchimol.
- 1a ed. - Buenos Aires: Fox Andina; Dalaga, 2011.
v. 17, 192 p. ; 19x15 cm. - (Desde cero; 19)
ISBN 978-987-1773-22-0

1. Informática. I. Daniel Benchimol, coord.
CDD 005.3

RedUSERS.com

Prólogo **al contenido**

Los microcontroladores generaron una revolución en la forma de pensar y diseñar circuitos electrónicos. Creados a mediados de la década del ochenta, rápidamente desplazaron a los microprocesadores en el campo del control industrial. Además, los microcontroladores poseen muchas ventajas respecto a la lógica cableada y a la lógica programada debido a que tienen bajo costo, alta inmunidad al ruido eléctrico y pequeño tamaño.

En los años noventa del siglo pasado, los microcontroladores entraron al campo de la electónica de consumo y llegaron para quedarse. Actualmente, todos nuestros aparatos electrónicos diarios los poseen, desde el lavarropas hasta el celular. Es difícil imaginar el mundo de hoy sin los microprocesadores.

Estos pequeños chips nos permiten resolver tareas muy complejas ya que poseen en su interior las unidades básicas de una computadora; debido a esto, los microcontroladores eran llamados en un principio microcomputadoras.

Dentro de cada microcontrolador encontramos la CPU, una memoria de programa, memoria de datos, el circuito de reset y el circuito oscilador, además de los puertos de entrada/salida. La CPU es el elemento principal de un microcontrolador, se conecta con los periféricos para conformar la estructura interna de estos. Luego de esta introducción, los invitamos a descubrir en profundidad el universo de los microcontroladores.

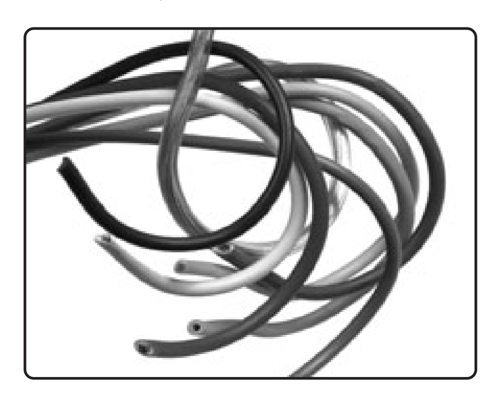
El libro **de un vistazo**

Este libro está enfocado en aquellas personas que quieran estudiar en profundidad el mundo de los microcontroladores. Empezamos con una introducción a las señales digitales y a la electrónica digital. Analizamos el desarrollo de los microprocesadores hasta llegar a los diferentes tipos de microcontroladores.



CAPÍTULO 1
SEÑALES ANALÓGICAS Y DIGITALES

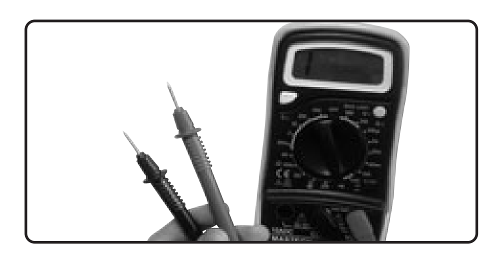
Conoceremos en detalle la diferencia entra las señales analógicas y las digitales. Estudiaremos el sistema binario y las compuertas lógicas. Aplicaremos los conceptos y elementos estudiados para armar un circuito que nos permitirá mantener un artefacto activado durante un tiempo.





CAPÍTULO 2
ELECTRÓNICA DIGITAL

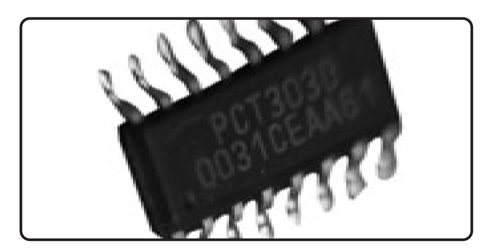
Estudiaremos la aritmética binaria. Conoceremos la diferencia entre los circuitos lógicos combinacionales y los secuenciales. Veremos los distintos tipos de estos últimos, como los asincrónicos y los sincrónicos. Analizaremos en detalle el funcionamiento de los registros.





CAPÍTULO 3
MEMORIAS

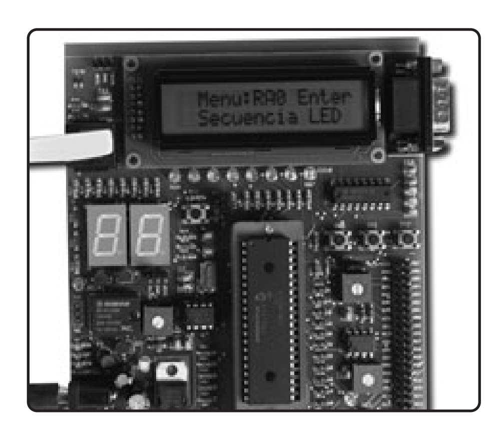
Analizaremos las diversas formas de almacenar información digital para procesarla oportunamente. Veremos la estructura interna de los diferentes tipos de memorias y sus características principales. Realizaremos un pequeño ejemplo simulado en ISIS.





CAPÍTULO 4
MICROPROCESADORES Y MICROCONTROLADORES

Aprenderemos el concepto de almacenamiento de datos y de instrucciones de programa para el procesador, sus diferencias y funciones dentro de un bloque computacional. Realizaremos un repaso de los distintos lenguajes de programación para microcontroladores.



CAPÍTULO 5
MICROCONTROLADOR PIC16F

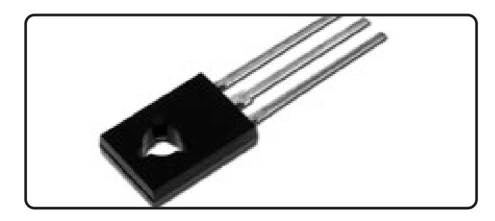
Aprenderemos a manejar los pines del microcontrolador para así poder operar sobre otros dispositivos, como los LEDs. Estudiaremos los espacios de memoria de esta familia, conoceremos las áreas de datos y de programa, y su función en la arquitectura.





CAPÍTULO 6
MICROCONTROLADOR PIC18F

Abordaremos las características principales de los microcontroladores PIC18F, que pertenecen a la familia de rango medio avanzado de los micros de 8 bits. Analizaremos la estructura interna de esta nueva familia de PICs.





SERVICIOS AL LECTOR

En este último apartado, encontraremos un índice temático que nos ayudará a encontrar de forma más rápida y precisa los principales conceptos de la obra.

Contenido **del libro**

CAPÍTULO 3 MEMORIAS 069

CAPÍTULO 4 MICROPROCESADORES Y MICROCONTROLADORES 091

CAPÍTULO 5 MICROCONTROLADOR PIC16F 119

CAPÍTULO 6 MICROCONTROLADOR PIC18F 145

SERVICIOS AL LECTOR 173

Introducción **a Microcontroladores**

El microprocesador es, tal vez, el mayor exponente del desarrollo de la electrónica digital, ya que puede ser programado para realizar las operaciones lógicas que veremos en los siguientes capítulos.

En el primer capítulo de este libro dedicado a los microcontroladores, descubriremos los primeros circuitos: las compuertas y su tratamiento, y veremos qué es posible construir con ellas. Además, estudiaremos cómo con el álgebra booleana y un sistema de numeración binario, se pueden realizar complejas operaciones lógicas o aritméticas sobre las señales de entrada.

Analizaremos en profundidad conceptos y dispositivos fundamentales en los cuales se basa la electrónica digital. También, conoceremos el funcionamiento de las memorias, sus diferentes tipos y aplicaciones.

Para profundizar más sobre el tema principal del libro, conoceremos la arquitectura de los microcontroladores PIC16F887, y veremos cómo crear nuestros primeros programas y manejar el entorno MPLAB. Además, abordaremos el microcontrolador PIC18LF4620, con el que aprenderemos a programar en C y realizar proyectos más avanzados.



Capítulo 1

Señales analógicas y digitales



Estudiaremos las señales analógicas y digitales, el sistema binario y las compuertas lógicas.

Sistemas **analógicos**

La representación discreta y binaria de las magnitudes ha permitido el desarrollo de la mayoría de los sistemas que operamos a diario y de los que operan por sí mismos, aun sin que nos demos cuenta. Estos sistemas se basan en el procesamiento de datos binarios, representados por valores discretos de tensión.

Por ejemplo, el microprocesador no es más que un gran conjunto de componentes elementales, como las compuertas lógicas. Estas, a su vez, son simples circuitos electrónicos como los que estudiamos aquí, en los cuales se explota alguna condición que permite obtener una respuesta acorde a una operación lógica. Se trata de una convención acerca de la representación de un concepto, operando sobre la representación binaria de una magnitud.

Los sistemas analógicos están relacionados con el mundo físico que nos rodea; son el mundo que experimentan nuestros sentidos. Estas magnitudes se presentan en forma continua, es decir que pueden tomar un número infinito de valores entre dos puntos de una escala graduada. Podemos mencionar muchos ejemplos, como la longitud de una columna de mercurio en un termómetro, una balanza de aguja y el instrumento de D'Arsonval o miliamperímetro de continua analógico. Apreciamos, entonces, que existe una relación inherente entre el mundo de los sentidos, lo analógico, el infinito y la idea de continuidad.

El término analógico proviene de la palabra **analogía** y viene a dar luz sobre el hecho de que, para medir magnitudes físicas de características inherentemente continuas, debemos recurrir a comparaciones o equivalencias, estableciendo ciertas convenciones o patrones de referencia. Por ejemplo: el kilo, el metro y el litro son patrones de referencia que, por analogía, nos dan una idea de la magnitud del fenómeno físico en estudio.

CIRCUITOS ANALÓGICOS

Los circuitos analógicos gobiernan y adoptan magnitudes físicas –como tensión, corriente, campo eléctrico y flujo magnético– para lograr un fin determinado. Por ejemplo, la amplificación de una señal eléctrica que excita un parlante o la conversión de niveles de tensión en un transformador, entre muchos otros casos.

Los circuitos analógicos gobiernan y adoptan magnitudes físicas



IDEA DE CONTINUIDAD

Dados dos puntos consecutivos sobre una recta, siempre es posible hallar uno intermedio, de la misma manera que entre dos números reales siempre existirá otro. Así, la idea de infinito queda asociada con la de continuidad.

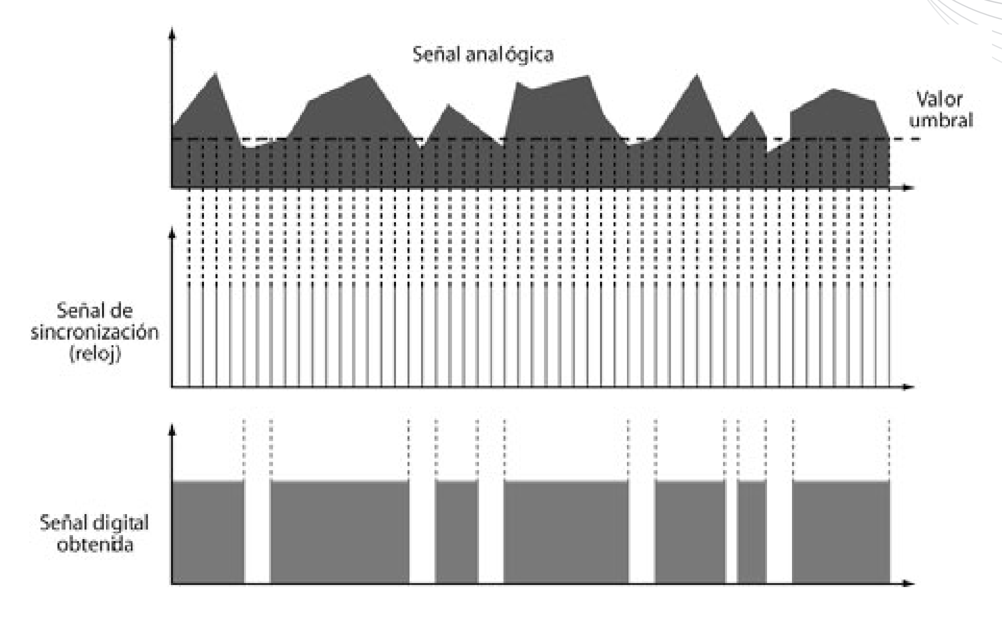


FIGURA 1. Observamos cómo actúa un conversor A/D, tomando muestras sincronizadas por reloj de una señal analógica.

La transmisión de información también es parte del mundo analógico, como las señales de **AM** y **FM** de radio. En ellas se transmite información aprovechando la naturaleza de la propagación de las ondas electromagnéticas, modulando una portadora en amplitud (AM) o frecuencia (FM) mediante técnicas puramente analógicas. La variación de la corriente de campo de un motor de continua para el control de su velocidad también es una señal que podemos denominar analógica.

SISTEMAS DIGITALES

Los sistemas electrónicos nunca son totalmente analógicos, pero tampoco estrictamente digitales; son híbridos. En este apartado, vamos a profundizar en los sistemas digitales.

El término digital proviene de **dígito**, sinónimo de dedo, y nos acerca al mundo de lo discreto, de lo que podemos contar; en definitiva, de lo **discontinuo**. En cierto sentido, no necesitamos los números reales para cuantificar un fenómeno, sino que nos alcanza con los números enteros. Debemos destacar que el hecho de que un sistema sea digital no implica que se trabaje estrictamente con números binarios. Un sistema digital puede tranquilamente ser de naturaleza decimal.

Lo que sucede es que **el sistema de numeración binario** se presta de manera excepcional para brindar soluciones a infinidad de cuestiones vinculadas a la ingeniería electrónica: desde lo estructural, pasando por lo matemático, hasta las ventajas logra-

En electrónica, todo sistema trabaja con señales digitales de naturaleza binaria

das en el almacenamiento, procesado, fiabilidad y transporte de información. Es por eso que el sistema de numeración binario se vuelve indispensable en el diseño, los dispositivos conversores A/D (analógico-digitales) y los conversores D/A (digitales/analógicos), ver **Figura 1**.

En electrónica, todo sistema trabaja con señales digitales de naturaleza binaria. Esto implica la presencia de solo dos estados posibles: conducción (ON) y corte (OFF), en equivalencia a verdadero y falso, nociones que maneja todo sistema lógico.

Los transistores de los circuitos integrados actúan como llaves de conmutación, al permitir la conducción o interrupción de un circuito eléctrico modificando el estado del sistema. Es decir, gobiernan el comportamiento lógico del circuito. Hablaremos entonces de transistores y tecnologías que se ajustarán a niveles de tensión representativos de dichos estados, ver **Tabla 1**.

VALORES	ESTADOS
5 V / 3,3 V / 1,8 V	ON (1 en binario o estado alto)
0 (cero) V	OFF (0 en binario o estado bajo)

TABLA 1. Todas las operaciones lógicas podrán efectuarse en estas condiciones.



LÓGICA COMBINACIONAL Y LÓGICA SECUENCIAL

En este punto, debemos hacer una distinción entre lógica combinacional y lógica secuencial. La **combinacional** se refiere a un sistema que reacciona siempre de la misma manera frente al mismo juego o combinación de entradas. Es decir, cada vez que se aplica una combinación de entradas determinada, el sistema devuelve el juego de salidas correspondiente. Esta es una operación sin memoria.

Un sistema **secuencial,** en cambio, tiene memoria. Responde no solo a una determinada combinación de entradas, sino que también coteja con algún resultado anterior para realizar una acción.

Es decir, depende del orden y de la secuencia con que se ejecutan las combinaciones a su entrada. Es importante aclarar que todo circuito digital puede desglosarse en dos grandes bloques: uno **combinacional** y otro **secuencial**, que actúan en conjunto (**Figura 2**).

Diagrama de bloque de un circuto sec uencial

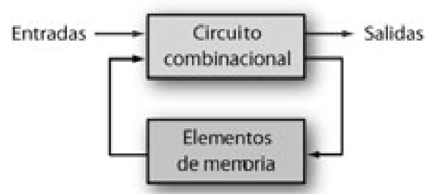


FIGURA 2. Vemos aquí la conformación de un sistema digital que integra módulos.

LOS SISTEMAS DE NUMERACIÓN

La necesidad de representar cantidades de un determinado objeto mediante símbolos ha llevado a desarrollar diversos métodos de representación. Analizaremos a continuación los sistemas de numeración actuales.
Históricamente, los esfuerzos se centraron en encontrar un sistema que precisara de la menor cantidad de símbolos para representar grandes cantidades, y que facilitara las operaciones y cálculos. Los sistemas posicionales de numeración surgieron en forma independiente, tanto en Oriente como en América. En ambos casos, hay un símbolo representante de la **ausencia de cantidad: el cero**.

Los sistemas posicionales utilizan un conjunto limitado y constante de símbolos, donde cada uno representa una cierta cantidad de unidades. Pero, además, dependiendo de la posición que ocupe en el grupo de caracteres de representación, este símbolo tendrá mayor o menor peso. Nuestro sistema decimal, por ejemplo, es un ejemplo típico de un sistema de **representación posicional**. Lo llamamos decimal pues, con la combinación de 10 dígitos, es posible representar cualquier cantidad: 0,1, 2, 3, 4, 5, 6, 7, 8 y 9. Veamos un ejemplo del peso de los símbolos de acuerdo con su posición. Podemos, entonces, descomponer la cantidad **18.127** de la siguiente forma:

18.127 = 1 x 10.000 + 8 x 1.000 + 1 x 100 + 2 x 10 + 7

El decimal es un sistema de representación posicional de base 10; el binario es de base 2 y el hexadecimal, de base 16



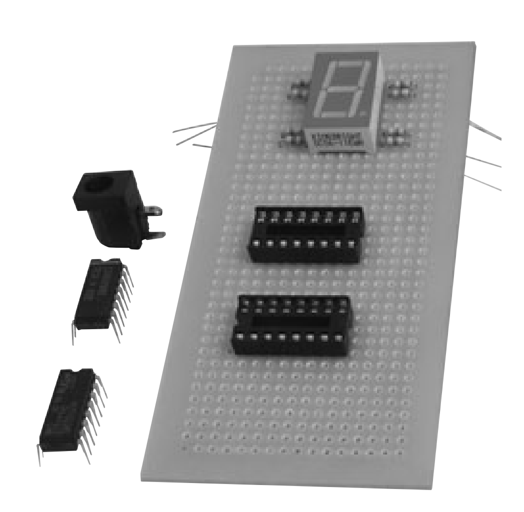
REGLA POSICIONAL

Para todo sistema entero de numeración posicional se cumple que el último dígito de la derecha representa unidades (**peso de valor 1**). El peso de cada posición se incrementa de derecha a izquierda en potencias de la base.

Observemos que cada dígito que conforma el número **18127** tiene un peso propio por la posición que ocupa en la cadena de caracteres. El peso de cada dígito en el sistema decimal es claramente múltiplo de **10**. Luego, la descomposición que sigue también puede lograrse:

$$\mathbf{18127 = 1 \times 1^{04} + 8 \times 1^{03} + 1 \times 1^{02} + 2 \times 1^{01} + 7 \times 1^{00}}$$

De aquí viene la denominación de sistema en **base 10**, sinónimo de sistema decimal. Este tipo de descomposición puede extenderse a los demás sistemas de numeración, como el binario que desarrollaremos más adelante.



Sistema **binario**

Estudiaremos en este apartado el sistema binario de numeración, utilizado en los sistemas digitales como base de operación matemática, almacenamiento y representación.

Se denomina sistema binario al sistema de numeración de base 2, que, como se desprende, consta de solo 2 dígitos para representar cantidades: 0 y 1. Con ellos podemos representar **22 = 4** valores; con 3 **dígitos, 23 = 8** valores; con 4 dígitos, **24 = 16** valores y así sucesivamente; por lo tanto, con N dígitos se podrá representar hasta **2N** valores. Es importante destacar que los ceros a la izquierda no cuentan, como es lógico, a la manera en que estamos acostumbrados en el sistema decimal. Para operar con números binarios, lo haremos intuitivamente del mismo modo que cuando utilizamos el sistema decimal, ver **Tabla 2**.

2 BITS	3 BITS	4 BITS	VALOR DECIMAL
00	000	000	0
01	001	0001	1
10	010	0010	2
11	011	0011	3
	100	0100	4
	101	0101	5
	110	0110	6
	111	0111	7
		1000	8
		1001	9
		1010	10
		1011	11
		1100	12
		1101	13
		1110	14
		1111	15

TABLA 2. **Los números binarios pueden formarse a partir de arreglos de distintas cantidad de dígitos.**

OPERACIONES CON NÚMEROS BINARIOS

En las operaciones típicas de suma, resta y multiplicación, se aplican las técnicas de acarreo, adaptadas en este caso a la operación con solo 2 símbolos (**Figura 3**). Esto es: **01 + 01**, en binario, arrojará el valor 10, ya que **1 + 1** no puede representarse con un solo símbolo. Se deja entonces un 0 en la posición menos significativa y se acarrea un 1 hacia la más significativa. El resultado es 10 binario (2 en decimal).

La multiplicación es todavía más intuitiva y se realiza de manera habitual. La única posibilidad de que una multiplicación entre 2 bits arroje 1 como resultado es que ambos sean 1. Es importante destacar que agregar un cero por derecha a un número binario tiene como resultado duplicar su valor. En efecto, dado el número 110 binario (6 decimal), el número 1100 (binario) corresponde al doble de su valor (12 en decimal).

REPRESENTACIÓN CON SIGNO

Es posible representar números binarios signados utilizando el bit más significativo como bit de signo: un 1 indicará negativo y un 0, positivo. Por ejemplo, como se ve en la **Tabla 3**, el número **1010** (10 decimal en binario **no signado**) representaría el número **-6 decimal en binario signado**.

Debido a que un bit se utiliza como signo, el número máximo que es posible representar para los positivos es **0111** (**7 decimal**). El mínimo negativo será el **1000** (**-8 decimal**), y el máximo negativo, el **1111** (**-1 decimal**).

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

TABLA 3. Representación de números binarios con signo.

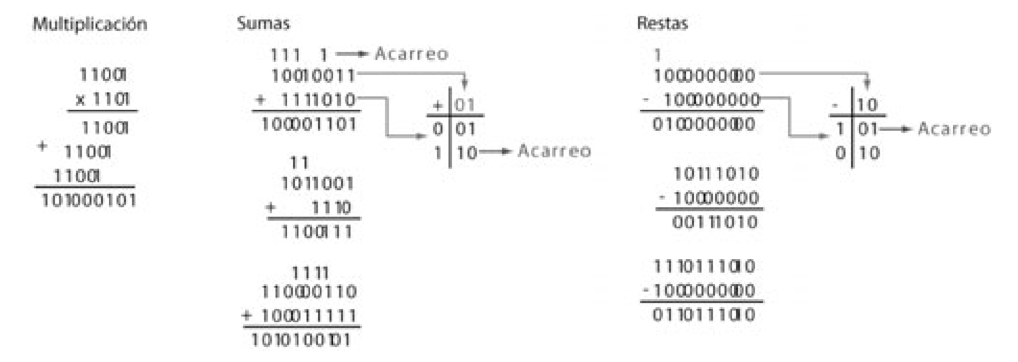


FIGURA 3. Observamos aquí las operaciones de suma, resta y multiplicación entre números binarios.

CONCEPTO DE MÓDULO

En representaciones de números binarios signados, se llama módulo al resultado de sumar la representación negativa de un número con su representación positiva. Por ejemplo, si sumamos **1111** (**-1 decimal**) a **0001** (**1 decimal**), obtendremos el módulo: **10000**. Se dice entonces que estos números son complementarios con respecto a su módulo.

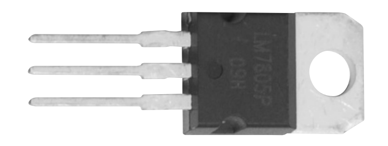
TRABAJO CON BINARIOS SIGNADOS

Cuando trabajamos con binarios signados, calculamos el módulo como el número binario correspondiente a la cantidad de combinaciones que se pueden lograr con la cantidad de bits utilizada incluyendo el signo. Por ejemplo, para el caso de utilizar 4 bits (incluido el signo), se puede lograr **24 = 16** combinaciones (de -8 a 7 pasando por el cero). Por lo tanto, el módulo de este arreglo de bits signados es 16; en binario: 10000.

Conociendo el módulo, es posible determinar la representación de números negativos, simplemente operando sobre la representación positiva, sin necesidad de tener a la vista todas las combinaciones posibles. Restando entonces al módulo la representación positiva, obtendremos la representación negativa de ese número con esa cantidad de bits (**Figura 4**).

```
  10000   (Módulo 24 = 16 en decimal)
-  0110   (Representación positiva: 6 decimal)
  -----
   1010   (Representación negativa: -6 decimal)
```

FIGURA 4. Cálculo de la representación negativa del número 0110 (6 decimal). Para módulo 10000 (24 = 16 decimal).



Se llama módulo al resultado de sumar la representación negativa de un número con su representación positiva

¿Por qué conocer **el sistema binario?**

Cuando hablamos de electrónica digital, nos referimos a sistemas electrónicos que procesan, almacenan, se comunican y operan en binario. Veremos aquí su importancia.

Un sistema digital de este tipo manejará internamente solo dos estados: 1 (alto) y 0 (bajo). En consecuencia, la organización de la información estará basada en arreglos de valores binarios. Surge entonces el concepto de **bit**, que no es más que el acrónimo de **binary digit** (dígito binario). Con un bit podremos representar dos estados o valores. El arreglo de 8 bits se denomina **byte**, término que deriva de la palabra anglosajona *bite*, "**mordisco**" en castellano. Se refiere a

la cantidad mínima de datos que un procesador puede "morder" a la vez, adoptándose por convención el tamaño de 8 bits. La traducción al español que toma la Real Academia es **octeto**, pero nosotros utilizaremos **byte** para evitar confusiones. De allí, las unidades de capacidad y almacenamiento más utilizadas:

- **Kilobyte = 1024 bytes**
- **Megabyte = 1024 Kb**
- **Gigabyte = 1024 MB**
- **Terabyte = 1024 GB**

Con un byte podemos manejar 256 valores posibles. Por ejemplo, en programación, se acostumbra definir variables de tipo carácter, de 1 byte de longitud. Se trabaja, así, con valores enteros en un rango de 0 a 255 (sin signo) o de -128 a 127 (con signo).
Es común también hablar de "words" o arreglos de 16 bits, no solo en programación de sistemas con microprocesadores y microcontroladores, sino también en algunos medios de almacenamiento, como memorias RAM y ROM, que proponen direccionamiento y palabras de datos mínimas de 16 bits. Por lo tanto, es una unidad de trabajo estándar que identifica una palabra de 2 bytes de longitud, muy utilizada para definir variables de tipo entero, capaces de manejar 65.536 valores distintos, suficientes para muchas de las operaciones más corrientes.

El sistema hexadecimal consta de 16 símbolos para representar números

DECIMAL	BINARIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

TABLA 4. **En la tabla se muestra una equivalencia entre valores decimales, binarios y hexadecimales.**



ALCANCE HEXADECIMAL

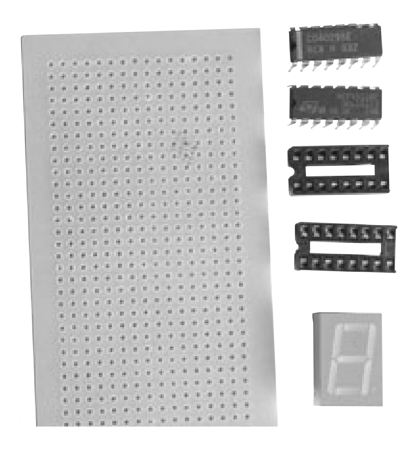
El sistema hexadecimal consta de **16** símbolos para representar números. Del **0** al **9** coinciden con los símbolos del sistema decimal. Luego, se agregan los caracteres **A**, **B**, **C**, **D**, **E** y **F** para obtener el juego de 16 símbolos, recordemos que **162 = 256**.

CONVERSIÓN DECIMAL A BINARIO

Para convertir un número de decimal a binario, hay que dividir el decimal sucesivamente por **2**, hasta obtener un cociente menor que el divisor. Este cociente (que será **0** o **1**, naturalmente), más los restos de las sucesivas divisiones efectuadas, constituye la representación binaria buscada. De esta forma, el cociente de la división será el dígito más significativo del número binario, y el menos significativo corresponderá al primer resto de la división (**Figura 5**).

```
18 |_2
 0   9 |_2
     1   4 |_2
         0   2 |_2
             0   1   = 10010b
```

FIGURA 5. Se muestra aquí el método de divisiones sucesivas para hallar la representación binaria de un número decimal.



CONVERSIÓN DECIMAL A HEXADECIMAL

Este método puede extenderse a conversiones de decimal a cualquier otro tipo de base, por ejemplo, hexadecimal. Para esto, habrá que dividir el número decimal sucesivamente por la base en la que se lo quiere representar; en nuestro caso, **16** (**Figura 6**).

```
5680 |_16
 88    355 |_16
  80    35    22 |_16
   0     3     6     1      = 1630h
```

FIGURA 6. El mismo método de divisiones sucesivas para hallar la representación ahora hexadecimal de un número decimal.

El orden de los dígitos del número hexadecimal obtenido es el inverso del que hemos obtenido en la división anterior.

CONVERSIÓN BINARIO A DECIMAL

Para realizar este procedimiento, hacemos el desarrollo en potencias de 2 de un número binario y sumamos los pesos:

$110101b = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 53d$

Es decir, en orden ascendente, el peso de cada dígito binario queda determinado: 1, 2, 4, 8, 16, 32, 64 (etcétera). Observamos cómo aumenta en potencias de 2, duplicando el peso en cada dígito. La conversión a decimal se realiza efectuando la suma de las contribuciones de peso de cada dígito que tome valor 1. En este caso, el número es 53.

CONVERSIONES DIRECTAS ENTRE BINARIO Y HEXADECIMAL

En los pasajes de binario a hexadecimal, y viceversa, se da una situación particular. Como una base es potencia de la otra, en nuestro caso **16 = 24**, es posible demostrar que los dígitos de la base menor (base 2 en nuestro caso) pueden agruparse en un número igual al exponente (4 en el ejemplo) para conformar un dígito de la base mayor.

BINARIO	CUARTETOS	HEXADECIMAL
1110100101	(0011) (1010) (0101)	3A5h
101111	(0000) (0010) (1111)	02Fh
101011100001	(1010) (1110) (0001)	AE1h

TABLA 5. Ejemplo de conversiones de binario a hexadecimal.

Para simplificar: partiendo un número binario en cuartetos (agregando ceros a la izquierda a la parte más significativa, si es necesario), el reemplazo de dicho cuarteto por el símbolo hexadecimal equivalente se realiza en forma directa (ver **Tabla 5**).

CONVERSIÓN INVERSA

La conversión inversa, de hexadecimal a binario, se lleva a cabo en forma directa y de la misma manera. Se reemplaza cada símbolo hexadecimal por un número binario de 4 bits. Por lo tanto, no se requiere ningún tipo de operación para los procedimientos de este tipo. Es por eso que el sistema hexadecimal es tan utilizado en sistemas electrónicos e informáticos. Para las personas resulta difícil trabajar en binario debido a que es fácil perderse con largas secuencias de unos y ceros para operar. El sistema hexadecimal permite representar de manera más compacta la información contenida en largas secuencias binarias que se almacenan y procesan en los equipos digitales.

El sistema hexadecimal es muy utilizado en sistemas electrónicos e informáticos

CÓDIGO BCD

Este tipo de codificación está diseñado con el objetivo de simplificar la conversión de decimal a binario, y evita la necesidad de recurrir a tediosas operaciones aritméticas, como en el método de divisiones sucesivas por 2.



SUMA DE BCD

Para sumar números codificados en **BCD**, simplemente operamos como si trabajáramos con números naturales. Si la suma parcial de un cuarteto supera el valor **1001** (**9** en decimal), se suma al resultado **0110** (**6**) y se acarrea **0001** al siguiente dígito BCD o cuarteto de bits.

De este modo, el **código BCD** no es más que la representación binaria con 4 bits de cada dígito de un número decimal, en forma individual. Las conversiones de BCD a decimal y de decimal a BCD se realizarán, entonces, directamente, por simple inspección: **9723d = (1001) (0111) (0010) (0011) BCD.** Sin embargo, este tipo de representación es una equivalencia; no es una conversión matemática como el traspaso de binario a hexadecimal o de hexadecimal a binario. No estamos representando números en ninguna base. Por lo tanto, la suma de los pesos de los bits individuales no arrojará el valor decimal representado en BCD (**Figura 7**).

```
          1 1 1
          0111     1001      79
        + 0011     0101    + 35
          1 1 1    1 1      114
          1010     1110
          0110     0110
  0001    0001     0101
    1       1        4
```

FIGURA 7. Vemos aquí la operación de suma de números BCD.

Las compuertas **lógicas**

Los circuitos lógicos digitales han revolucionado la electrónica, ya que son mucho más inmunes al ruido eléctrico, más rápidos y más versátiles que su contraparte analógica.

El continuo avance de la tecnología permite la aplicación de la electrónica digital en cada vez más áreas de especialización. Aquí veremos los bloques constitutivos de los circuitos digitales: las **compuertas lógicas**.

Un circuito lógico puede representarse con un mínimo de detalle, como una caja negra con una determinada cantidad de entradas y salidas. Entendemos por caja negra a un circuito del cual no tenemos datos sobre su composición interna, pero sobre el que sí podemos conocer cómo es su salida ante una cierta entrada. Como las entradas al circuito lógico pueden tomar solo los valores discretos **0** y **1**, la lógica llevada a cabo por él puede describirse completamente a través de una tabla que ignora su comportamiento eléctrico y define la salida con **0** y **1** (**Figura 8**).

VALORES LÓGICOS

Los valores lógicos se representan con un **0** o un **1**, y se denominan **dígitos binarios** o bits. Cada uno representa un rango de validez para una señal analógica. Un circuito lógico, cuya salida depende solo

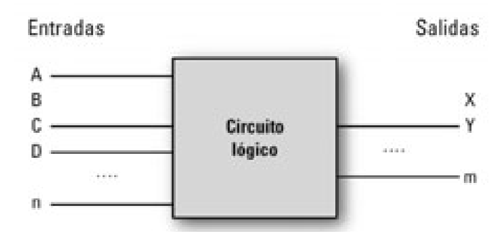


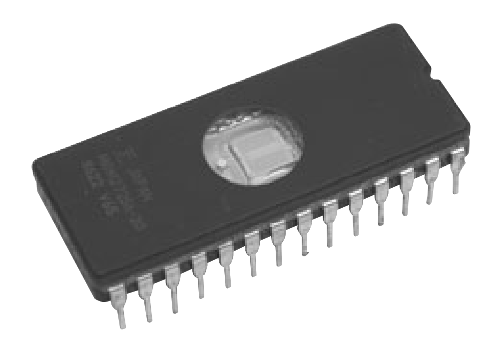
FIGURA 8. Representación de un circuito lógico. Las n **entradas aplicadas producen las** m **salidas. Desde este punto de vista, no resulta necesario conocer su estructura interna.**

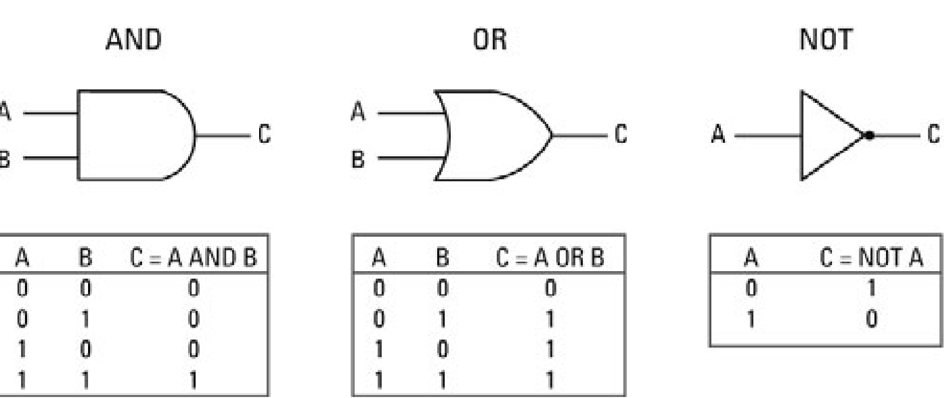
de sus entradas presentes, se conoce como **circuito combinacional,** y su funcionamiento se representa por medio de una **tabla de verdad**.

Cualquier circuito combinacional puede construirse sobre la base de tres compuertas lógicas fundamentales, denominadas **AND**, **OR** y **NOT**. Existe otra compuerta particular llamada **BUFFER**, en donde la señal lógica no sufre ningún cambio, es decir que la tensión de salida sigue a la de entrada. Estas compuertas se utilizan generalmente para regenerar señales débiles y convertirlas otra vez en señales fuertes para que puedan ser transmitidas a lo largo de cierta distancia sin pérdida de información.

En la **Figura 9** observamos la representación gráfica de la compuerta **NOT.** En la punta del triángulo se ha colocado un círculo que denota el carácter inversor de la función, e implica que el valor lógico presente en la entrada de la compuerta se invierte a la salida. De hecho, a la compuerta **NOT** también se la conoce con el nombre **inversor**. Si la entrada al inversor es **A**, entonces este implementa la función **NOT A** y se representa con $\bar{A}$ (nombre de la entrada con una raya en la parte superior).

¿Qué sucede si en cada entrada de la función **OR** colocamos un inversor? La configuración lógica que se obtiene se conoce con el nombre **NAND** y es la implementación inversa de la función **AND** (**Figura 10**).



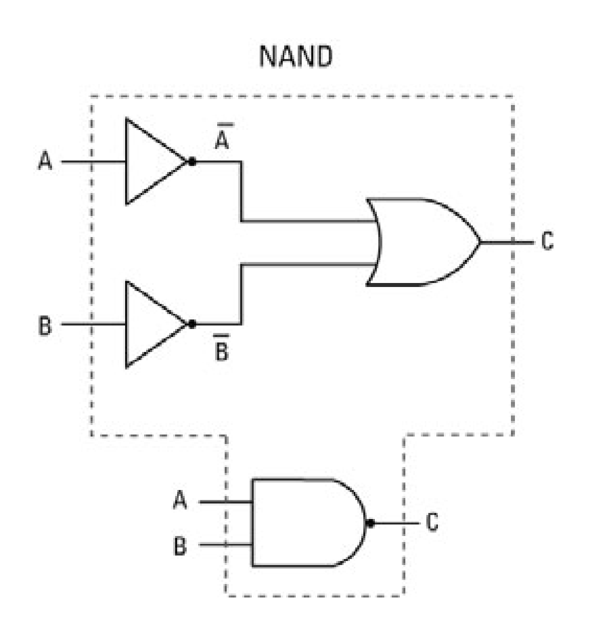


A	B	C = A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	C = A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	C = NOT A
0	1
1	0

FIGURA 9. Compuertas lógicas elementales AND, OR **y** NOT **y compuerta BUFFER. Su representación gráfica y su tabla de verdad, a partir de las cuales es posible construir cualquier circuito combinacional.**



A	B	C = A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

FIGURA 10. Compuerta lógica NAND. Implementación como compuerta OR con sus entradas negadas y su tabla de verdad. La compuerta NAND produce el resultado inverso de la compuerta AND.

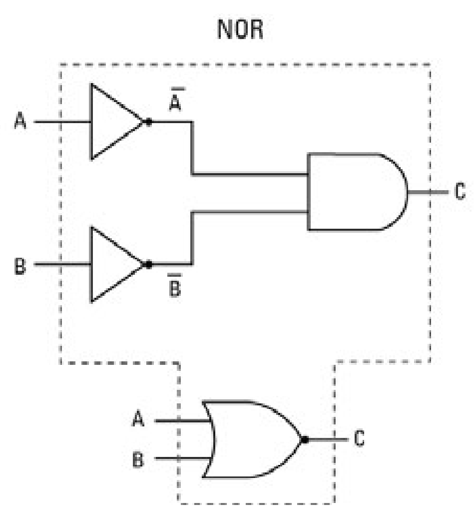
Ahora, veamos qué obtenemos si usamos la compuerta **AND** y colocamos inversores en sus entradas. La configuración lógica resultante se conoce como **NOR** y es la implementación inversa de la función **OR** (**Figura 11**).

Observemos la configuración lógica que se muestra en la **Figura 12**. Ésta se conoce como **OR-Exclusiva** o **XOR**, y su característica es que produce una salida lógica **1** cuando sus entradas son diferentes, mientras que arroja una salida lógica **0** cuando sus entradas son iguales.

DE LA ELECTRÓNICA A LA LÓGICA

Dijimos que un circuito lógico puede representarse como una caja negra con entradas y salidas. Esta caja negra es la que contiene un circuito electrónico que implementa la compuerta lógica representada por el bloque combinacional. En las páginas siguientes veremos algunos circuitos electrónicos muy básicos que implementan las compuertas lógicas elementales vistas anteriormente





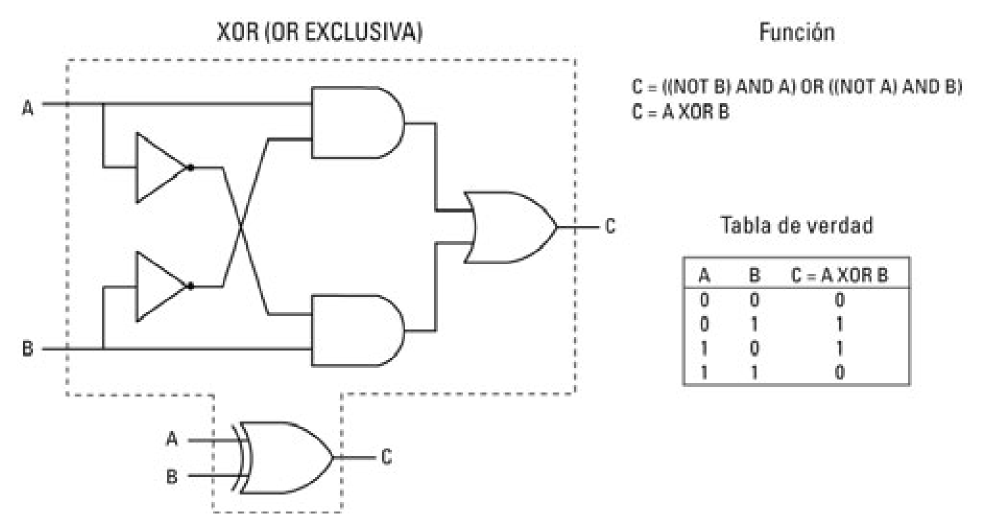
Función

C = (NOT A) OR (NOT B)
C = A NOR B

Tabla de verdad

A	B	C = A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

FIGURA 11. **Compuerta lógica NOR. Implementación como compuerta AND con sus entradas negadas y su tabla de verdad. La compuerta NOR produce el resultado inverso de la compuerta OR.**



A	B	C = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

FIGURA 12. **Compuerta lógica XOR. Implementación como secuencia AND-OR y su tabla de verdad. Se utiliza para identificar cuando dos entradas son iguales o distintas entre sí.**

En el circuito de la **Figura 13**, si en cualquier entrada (**A**, **B**) se presenta un valor de tensión positiva que haga conducir al diodo, este valor se observa a la salida (**S**). En caso contrario, el resistor fuerza 0 V. Entonces, dado que cualquier entrada que esté en **1** ocasiona un **1** a la salida, esta es una compuerta **OR**, pues esta característica se corresponde con la tabla de verdad de dicha compuerta.

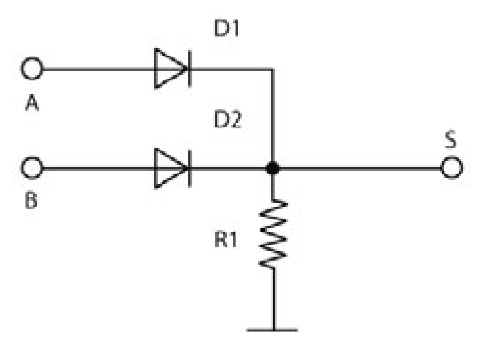


FIGURA 13. Una compuerta OR elemental con diodos y resistores (RDL o Resistor Diode Logic).

En el circuito de la **Figura 14**, si cualquier entrada (**A**, **B**) se conecta a 0 V, este valor se observa a la salida (**S**). En caso contrario, el resistor fuerza la tensión de alimentación. Entonces, dado que cualquier entrada que esté en **0** ocasiona un **0** a la salida, esta es una compuerta **AND**, pues esta característica se corresponde con la tabla de verdad de dicha compuerta.

Las compuertas lógicas implementan las funciones lógicas elementales

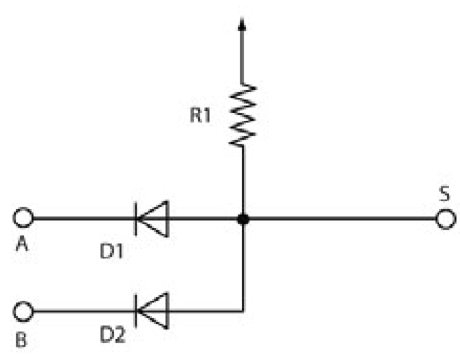


FIGURA 14. Una compuerta AND **elemental con diodos y resistores (**RDL **o Resistor Diode Logic).**

El circuito de la **Figura 15** utiliza un transistor tipo **NPN**, **Q1,** para implementar la compuerta **NOT**. En este caso, con una tensión de **1** lógico en el punto **A,** el transistor entra en saturación, y el punto **C** se coloca a la tensión de **0** lógico. Cuando el transistor entra en corte a través de la aplicación de una tensión de **0** lógico en el punto **A**, el punto **C** queda a tensión de fuente menos la caída de tensión en el resistor **R1**.

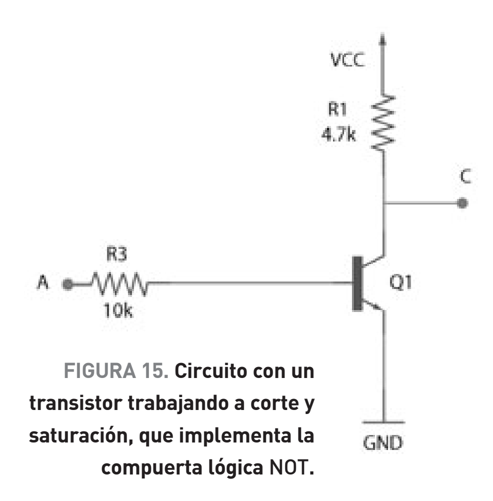


FIGURA 15. Circuito con un transistor trabajando a corte y saturación, que implementa la compuerta lógica NOT.

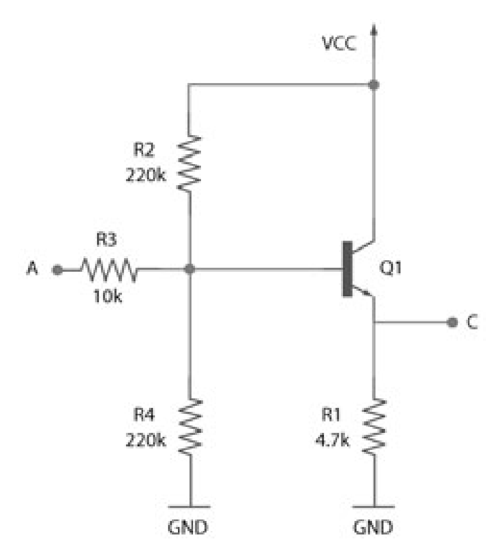


FIGURA 16. Circuito con un transistor, que implementa la compuerta BUFFER.

En la **Figura 16** observamos un circuito que implementa la compuerta BUFFER o seguidor de tensión. Cuando en el punto **A** tenemos una tensión correspondiente a un **0** lógico, el transistor **Q1** está en corte, y en el punto **C** tendremos una tensión equivalente a un **0** lógico. Cuando en **A** tenemos un **1** lógico, **Q1** conduce y coloca en el punto **C** a la tensión de fuente o **1** lógico.

En la **Figura 17** se presenta la implementación de la compuerta **NOR** agregando un inversor (basado en un transistor PNP) en cascada con la compuerta **OR** que ya vimos en la **Figura 13**.

La **Figura 18** muestra la implementación de la compuerta **NAND** agregando un inversor (esta vez, basado en un transistor PNP) en cascada con la compuerta **AND** vista en la **Figura 14**.

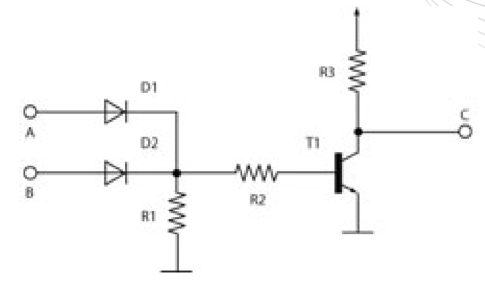


FIGURA 17. Circuito elemental que implementa la compuerta lógica NOR.

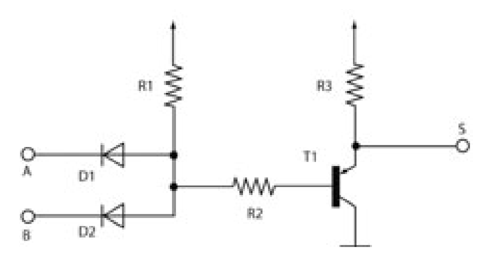


FIGURA 18. Circuito elemental que implementa la compuerta lógica NAND.

BUFFERS DE TRES ESTADOS

El diseño electrónico de las salidas de algunos dispositivos **CMOS** o **TTL** puede estar en un estado lógico **0**, **1** o de **alta impedancia,** también llamado **Hi-Z**. En la representación gráfica de un BUFFER de tres estados (**Figura 19**), se distinguen con claridad las entradas y las salidas. Adicionalmente, se representa una señal que actúa sobre el BUFFER, denominada **habilitación de salida,** que puede ser activo alto o bajo, dependiendo de si está presente o no el círculo que denota inversión. Cuando esta entrada está activa, el dispositivo se comporta como un BUFFER normal, mientras que si está negada, entonces la salida del BUFFER entra en un estado de alta impedancia y, funcionalmente, se comporta como si no estuviera allí.

La utilidad de estas compuertas es que permiten a múltiples fuentes compartir una sola línea de comunicación, mientras que solo una de ellas transmite datos por vez. Es decir, cuando un dispositivo quiere colocar información en la línea, deberá salir de su estado **Hi-Z** y empezar la transmisión, pero antes de hacerlo, debemos asegurarnos de que los demás dispositivos en la línea ingresaron en su estado **Hi-Z**; de lo contrario, habrá colisión de datos.

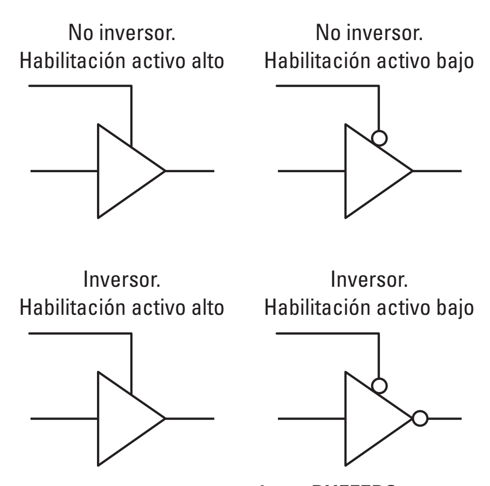


FIGURA 19. Representación de BUFFERS de tres estados, inversores y no inversores, cada uno con su respectiva habilitación activo alto o bajo.

COMPUERTAS A COLECTOR ABIERTO

La salida de compuertas **TTL** a colector abierto es otra de las configuraciones que pueden tener los dispositivos pertenecientes a esta familia. Se logra a través de una modificación interna de la compuerta TTL básica, que permite poner el colector del transistor de salida al alcance del usuario. Es decir, la salida de la compuerta es el colector de uno de los transistores con los que esta se encuentra construida. De esta manera, entonces, para lograr el correcto funcionamiento de la salida, es necesario colocar externamente un resistor de carga. Este requiere de un punto de referencia de tensión, que no necesariamente tiene que coincidir con la alimentación de la compuerta. El principal objetivo que se busca en este tipo de configuraciones es obtener un mayor nivel de corriente que pueda manejar la compuerta (**Figura 20**).

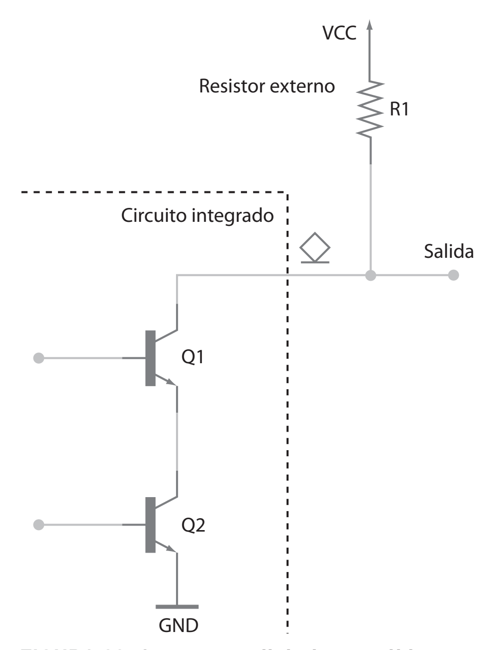


FIGURA 20. Compuerta digital con salida a colector abierto. Desde el exterior del circuito integrado es posible acceder al colector del transistor de salida. Cabe observar la notación para compuertas con salidas de este tipo: un rombo con una raya horizontal en su parte inferior.

Asimismo, y como el resistor externo puede estar conectado a un punto de tensión diferente de la alimentación de la compuerta, este tipo de dispositivos es ampliamente utilizado en aplicaciones en las que se necesita vincular dos familias lógicas que tienen umbrales diferentes (**CMOS** y **TTL**, por ejemplo).

También, como es posible manejar mayor potencia en forma directa desde la compuerta, se puede usar estos dispositivos para controlar pequeñas cargas, tales como lámparas, LEDs y relays. Otra aplicación de las compuertas a colector abierto es en lógica cableada, en donde la salida de varias compuertas se conecta a un resistor externo conectado a la tensión de fuente.

RESISTORES DE PULL-UP

Los resistores de pull-up se utilizan en circuitos lógicos digitales y se colocan en las entradas de los dispositivos lógicos. Su misión es asegurar que dichas entradas mantengan siempre un nivel lógico correcto y definido, para evitar que la entrada quede flotando. Una entrada flotante provoca un inadecuado funcionamiento de la compuerta y ofrece a la entrada propiamente dicha un nivel de tensión indefinido. Los resistores de **pull-up** elevan la tensión de la entrada donde están conectados a un determinado nivel, que suele ser la tensión de fuente (**Figura 21**).

El resistor propiamente dicho, sin embargo, debe tener un valor que haga débil la línea, en el sentido de

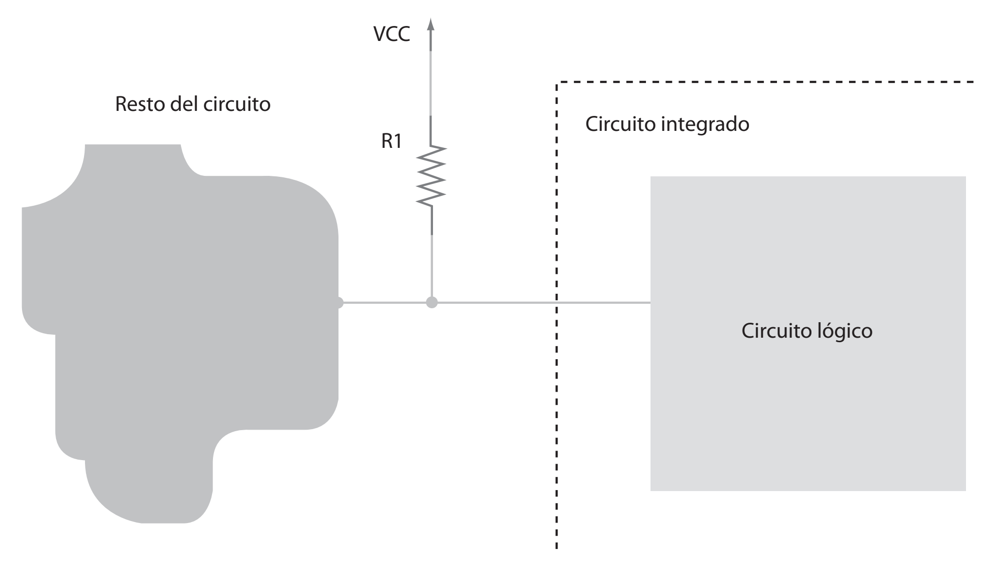


FIGURA 21. En esta figura podemos ver cómo los resistores de pull-up se colocan en las entradas para garantizar un nivel lógico definido en ellas.

que si otro dispositivo trata de imponer un nivel de tensión distinto en ella, el pull-up no se va a resistir y cederá sin inconvenientes.

La función principal de los resistores de pull-up es prevenir un exceso de corriente en el circuito, que ocurriría si un dispositivo tratara de llevar un punto a un determinado nivel de tensión cuando este ya tuviera uno distinto.

Así como existen los resistores de pull-up, también están los de **pull-down**, que son idénticos a los primeros, excepto que en vez de elevar la tensión de una entrada lógica a cierto nivel de tensión, la bajan a nivel de tierra o masa.

Los resistores de pull-up generalmente consumen menos potencia que los de pull-down. Por este motivo, son preferidos en los circuitos digitales donde la potencia consumida suele ser un tema crítico a la hora de diseñar.

LÓGICA CABLEADA

Se conoce con este nombre a las conexiones que implementan compuertas lógicas mediante la conexión directa de dispositivos de colector abierto o equivalente (drenaje abierto en **MOS**). Cuando cualquiera

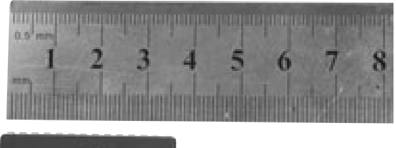


de los transistores está en conducción, el bus se encuentra a nivel lógico **0**. Solo cuando todos estén al corte, obtendremos la tensión de la fuente de alimentación a través del resistor **R1**. Dependiendo de la lógica que activa al transistor de colector abierto, activo alto o activo bajo, esta conexión se conoce como **wired-AND** (**AND cableada**) o **wired-OR** (**OR cableada**), respectivamente.

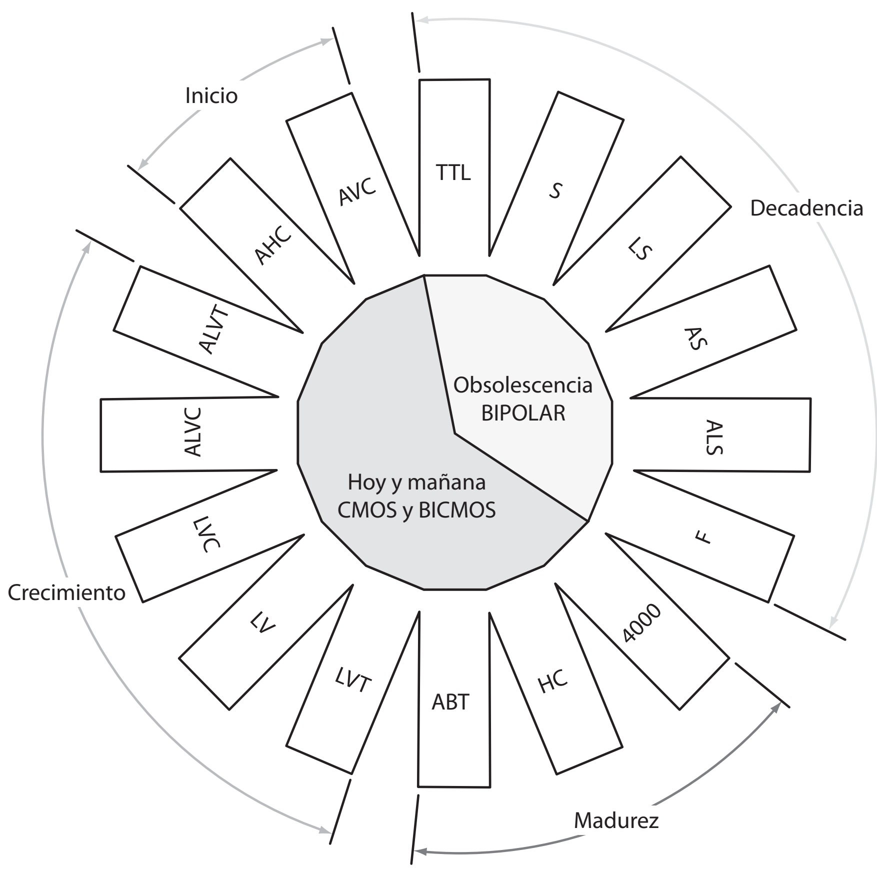
FAMILIAS LÓGICAS, NIVELES Y UMBRALES

Con el objetivo de lograr mejores prestaciones en los circuitos lógicos digitales, se viene dando una constante evolución que da origen a las distintas **familias lógicas** (**Figura 22**). Dentro de ellas, hay diversas subfamilias con características distintivas. Esta evolución que va experimentando cada una de las tecnologías pasa por varias etapas: **inicio** y **crecimiento**, **madurez** y **decadencia** o desuso. Pero siempre el objetivo buscado por cualquiera de estas tecnologías es **reducir el consumo** e **incrementar la velocidad de conmutación**.

Los resistores de pull-up se colocan en las entradas de dispositivos lógicos para forzar el estado alto







TTL	Transistor-Transistor Logic	ABT	Tecnología BICMOS avanzada
S	TTL Schottky	LVT	BICMOS de baja tensión
LS	TTL Schottky bajo consumo	LV	Baja tensión
AS	TTL Schottky mejorada	LVC	CMOS de baja tensión
ALS	Versión mejorada de LS	ALVC	CMOS de baja tensión mejorada
F	TTL de alta velocidad	ALVT	BICMOS de baja tensión mejorada
4000	Serie CMOS	AHC	CMOS de alta tensión mejorada
HC	CMOS de alta velocidad	AVC	CMOS de muy baja tensión mejorada

FIGURA 22. **Representación de familias lógicas en etapas de inicio, crecimiento, madurez y decadencia. Vale observar que la tecnología bipolar está pasando su etapa de madurez y tiende al desuso.**

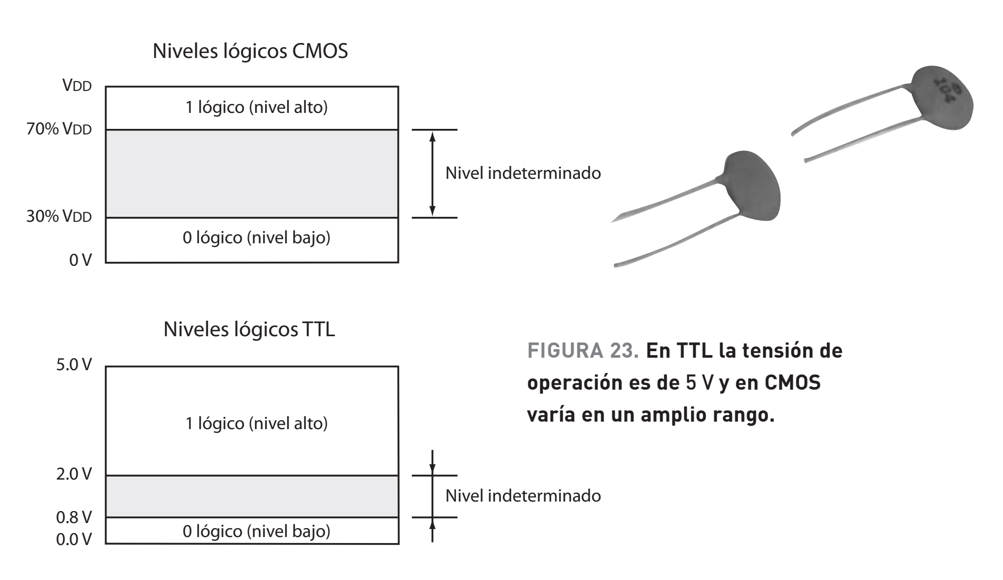


FIGURA 23. En TTL la tensión de operación es de 5 V y en CMOS varía en un amplio rango.

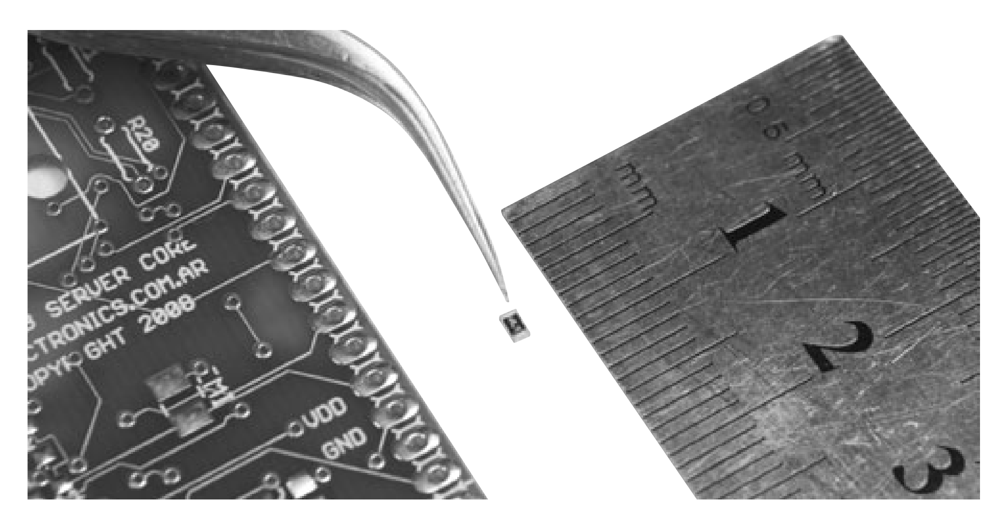
SUBFAMILIA	TIEMPO DE PROPAGACIÓN (NSEG)	VELOCIDAD DE CONMUTACIÓN (MHZ)	CONSUMO DE POTENCIA POR COMPUERTA (MW)
TTL estándar (54/74)	10	35	10
TTL de bajo consumo (54L/74L)	33	3	1
TTL de alta velocidad (54H/74H)	6	50	22
TTL Schottky (54S/74S)	3	125	20
TTL Schottky de bajo consumo (54LS/74LS)	10	45	2

TABLA 6. Alguno de los integrantes más importantes de la familia TTL, con sus características de velocidad de conmutación, tiempos de propagación y consumo de potencia.

FAMILIA CMOSSUBFAMILIA	TIEMPO DE PROPAGACIÓN (NSEG)	CONSUMO DE POTENCIA POR COMPUERTA @ 1 MHZ (MW)
Serie 4000	35	0,60
CMOS de alta velocidad (HC)	35	0,06
CMOS de alta velocidad compatible con TTL (HCT)	35	0,06
CMOS avanzado (AC)	35	0,75
CMOS avanzado compatible con TTL (ACT)	35	0,75

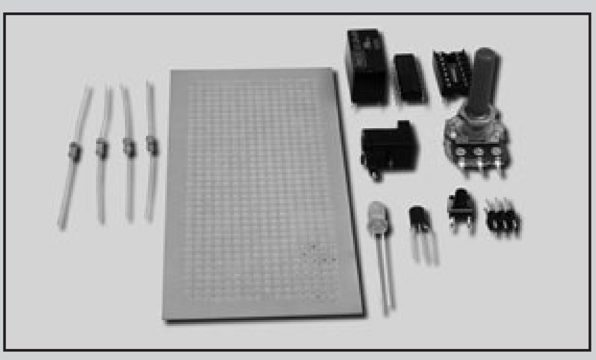
TABLA 7. Algunos de los integrantes más importantes de la familia CMOS, con sus características de consumo de potencia y tiempos de propagación.

Podemos definir una familia lógica como una estructura base a partir de la cual es posible construir diversas arquitecturas lógicas. Dicha estructura base involucra a todos los componentes con los que están constituidas las compuertas lógicas. Las arquitecturas lógicas a las que nos referimos están formadas por elementos discretos, tales como transistores, resistores y diodos, entre otros. Como estamos hablando de electrónica digital, no debemos descuidar el hecho de que las señales pueden tomar dos estados bien definidos: **alto** o **bajo**.



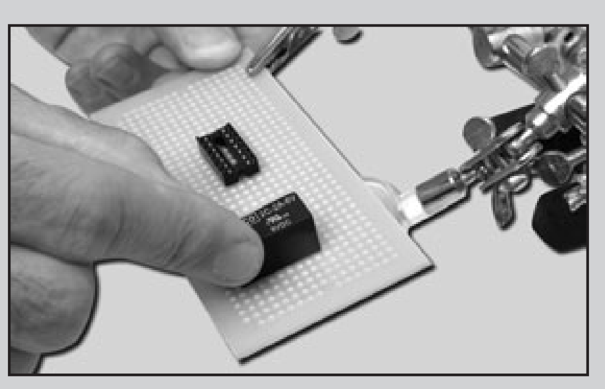
PASO A PASO /1
Construir un temporizador con 4093





Debe disponer de todos los componentes necesarios para el armado del circuito, verificando sus valores con el circuito esquemático. Asimismo, utilice las herramientas adecuadas para facilitar la inserción y posterior soldadura de todos ellos: un soldador con punta cerámica, una pinza de punta fina y un alicate resultarán de gran utilidad.

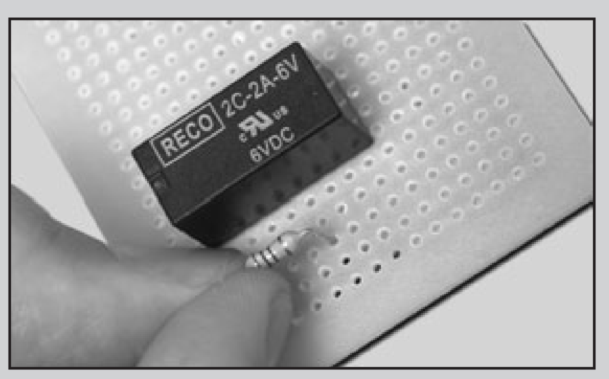




Presente en la placa los componentes más grandes para lograr su mejor ubicación, verificando el circuito (deben estar separados unos de otros). Una vez ubicados, debe soldar los pines por la parte de atrás de la placa, cuidando siempre que el componente quede al ras y nunca elevado de la placa. Aún no coloque el circuito integrado.

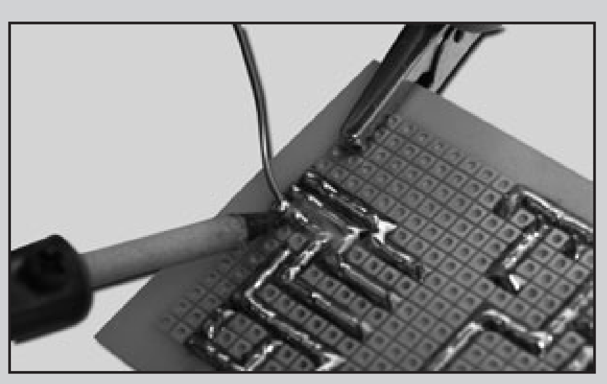
PASO A PASO /1 (cont.)





Una vez ubicados los componentes más grandes, continue con la disposición de los más pequeños. Solde los resistores, el transistor, el capacitor, el LED, el pulsador y el conector de pines. Utilice la pinza de punta para doblar y acomodar los alambres de los componentes de manera que quede prolijo, y el alicate, para cortar los alambres.

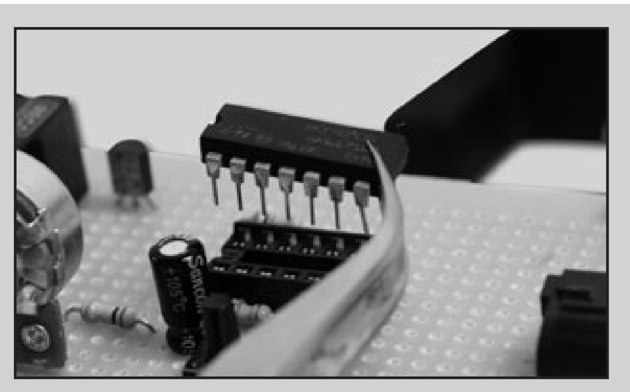




Debe observar el circuito y realizar las uniones entre los componentes con cables o con estaño, como en este caso. Es preciso cuidar que las pistas de estaño queden bien unidas entre sí y que no haya soldaduras "frías" o mal hechas, porque esto puede hacer que no conduzcan. También conviene evitar el exceso de estaño y la unión entre pistas diferentes.

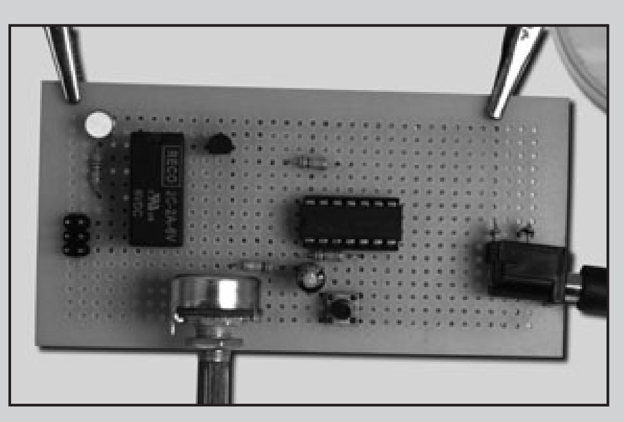
PASO A PASO /1 (cont.)





Utilice una pinza de Bruselas para colocar el circuito integrado en el zócalo correspondiente. En este punto, debe prestar atención a la ubicación del pin n° 1 del zócalo y del circuito integrado. Verifique que los componentes estén bien soldados y utilice un multímetro para comprobar que no haya continuidad entre las pistas de alimentación (VCC y GND).





Conecte el circuito a la fuente de alimentación. Según el relé que utilice, puede ser 5 V o 12 V. Hay que utilizar un relé acorde; es recomendable uno de 12 V, de los empleados en paneles de alarma. Lleve el potenciómetro P1 a su posición media y presione el pulsador para dar inicio al ciclo de temporización.

Por lo tanto, los circuitos que componen las compuertas lógicas deben tener dos regiones de operación bien establecidas y diferenciadas entre sí.

Las familias lógicas pueden clasificarse de acuerdo con el tipo de elemento semiconductor en el cual se basan. Tenemos, entonces, **familias bipolares,** que utilizan transistores bipolares, diodos y resistores; y **familias MOS,** que emplean transistores de efecto de campo MOS. Entre otras características, estas familias difieren en los rangos de tensión para los cuales se definen los niveles lógicos de cada una. En la **Figura 23** se observa esta diferencia.

Las familias bipolares se basan en transistores de unión o bipolares. La más representativa y de mayor uso es la **TTL,** dentro de la cual hay diferentes subfamilias que describiremos a continuación:

TTL (Transistor-Transistor Logic)

Introducida en los años 60, esta fue la familia más utilizada en dispositivos **SSI** y **MSI** (pequeña y mediana escala de integración, respectivamente), y en la actualidad está siendo desplazada por la **CMOS**. Existen diversas subfamilias **TTL**, cada una de las cuales tiene una característica particular: **S** (Schottky), **LS** (Schottky de bajo consumo), **AS** (Schottky mejorado), **ALS** (Schottky mejorado de bajo consumo), **F** (alta velocidad) y **L** (bajo consumo), entre las más relevantes, ver **Tabla 6**.

Familias MOS

Utilizan transistores de efecto de campo como elementos de conmutación. La más representativa de la familia es la CMOS, dentro de la cual existen diferentes subfamilias descriptas a continuación:



CMOS (Complementary Metal Oxide Semiconductor): actualmente está desplazando a la familia TTL en dispositivos **SSI** y **MSI** (pequeña y mediana escala de integración, respectivamente), debido a sus superiores características de velocidad, potencia disipada, márgenes de ruido y fanout. Existen diversas subfamilias CMOS, cada una de ellas con una característica especial: **HC** (alta velocidad), **AHC** (alta velocidad avanzado), **AC** (avanzado), **HCT** (alta velocidad compatible con TTL) y **FACT** (alta velocidad avanzado compatible con TTL), entre las más relevantes, ver **Tabla 7**.

TEMPORIZADOR CON 4093

Luego de haber leído abundante teoría, en el **Paso a paso 1** describimos los pasos para construir un temporizador. El **CD4093B** pertenece a la familia **CMOS** y contiene cuatro compuertas **NAND** de dos entradas cada una, del tipo **Schmitt-Trigger**. Esto permite fijar dos umbrales bien definidos para el cambio de estado. Según vemos en el circuito, el circuito en reposo mantiene a IC1B en 1 mediante la realimentación, con el capacitor descargado. El relé mantiene NA abierto y NC cerrado con COM. Al presionar el pulsador, IC1A cambia a **1**, y esto pone a IC1B en **0**, que mediante la realimentación mantiene a IC1A en **1**, hasta que se cargue el capacitor. Cuando esto ocurre, IC1B cambia otra vez a **1**, poniendo a IC1A en **0** y el capacitor se descarga.

Mientras el ciclo dure, el transistor **T1** estará polarizado y conduciendo, el **LED** se encenderá y el relay cerrará NA con COM.

Multiple **choice**

1 ¿Qué base tiene el sistema de representación posicional decimal?

a- 2.
b- 5.
c- 10.
d- 16.

2 ¿Cuál de los siguientes no es un sistema de representación posicional?

a- Decimal.
b- Binario.
c- Hexadecimal.
d- Ninguno de los anteriores.

3 ¿Cuál de los siguientes sistemas se utiliza de base en los sistemas digitales?

a- Decimal.
b- Binario.
c- Hexadecimal.
d- Ninguno de los anteriores.

4 ¿Cuántos símbolos tiene el sistema hexadecimal para representar números?

a- 2.
b- 5.
c- 10.
d- 16.

5 ¿Cuál de las siguientes unidades de almacenamiento es mayor?

a- Terabyte.
b- Gigabyte.
c- Megabyte.
d- Kilobyte.

6 ¿Cuál de las siguientes no forma parte de las tres compuertas lógicas fundamentales de un circuito combinacional?

a- If.
b- And.
c- Or.
d- Not.

Respuestas: 1 c, 2 d, 3 b, 4 d, 5 a, 6 a.

Capítulo 1

Señales analógicas y digitales



Estudiaremos las señales analógicas y digitales, el sistema binario y las compuertas lógicas.

Sistemas **analógicos**

La representación discreta y binaria de las magnitudes ha permitido el desarrollo de la mayoría de los sistemas que operamos a diario y de los que operan por sí mismos, aun sin que nos demos cuenta. Estos sistemas se basan en el procesamiento de datos binarios, representados por valores discretos de tensión.

Por ejemplo, el microprocesador no es más que un gran conjunto de componentes elementales, como las compuertas lógicas. Estas, a su vez, son simples circuitos electrónicos como los que estudiamos aquí, en los cuales se explota alguna condición que permite obtener una respuesta acorde a una operación lógica. Se trata de una convención acerca de la representación de un concepto, operando sobre la representación binaria de una magnitud.

Los sistemas analógicos están relacionados con el mundo físico que nos rodea; son el mundo que experimentan nuestros sentidos. Estas magnitudes se presentan en forma continua, es decir que pueden tomar un número infinito de valores entre dos puntos de una escala graduada. Podemos mencionar muchos ejemplos, como la longitud de una columna de mercurio en un termómetro, una balanza de aguja y el instrumento de D'Arsonval o miliamperímetro de continua analógico. Apreciamos, entonces, que existe una relación inherente entre el mundo de los sentidos, lo analógico, el infinito y la idea de continuidad.

El término analógico proviene de la palabra **analogía** y viene a dar luz sobre el hecho de que, para medir magnitudes físicas de características inherentemente continuas, debemos recurrir a comparaciones o equivalencias, estableciendo ciertas convenciones o patrones de referencia. Por ejemplo: el kilo, el metro y el litro son patrones de referencia que, por analogía, nos dan una idea de la magnitud del fenómeno físico en estudio.

CIRCUITOS ANALÓGICOS

Los circuitos analógicos gobiernan y adoptan magnitudes físicas –como tensión, corriente, campo eléctrico y flujo magnético– para lograr un fin determinado. Por ejemplo, la amplificación de una señal eléctrica que excita un parlante o la conversión de niveles de tensión en un transformador, entre muchos otros casos.

Los circuitos analógicos gobiernan y adoptan magnitudes físicas



IDEA DE CONTINUIDAD

Dados dos puntos consecutivos sobre una recta, siempre es posible hallar uno intermedio, de la misma manera que entre dos números reales siempre existirá otro. Así, la idea de infinito queda asociada con la de continuidad.

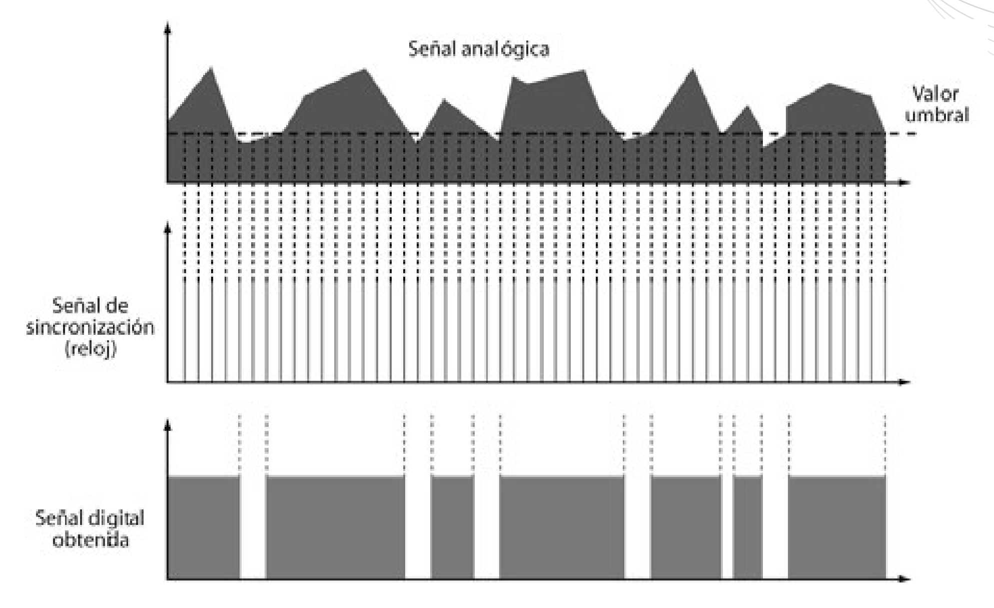


FIGURA 1. Observamos cómo actúa un conversor A/D, tomando muestras sincronizadas por reloj de una señal analógica.

La transmisión de información también es parte del mundo analógico, como las señales de **AM** y **FM** de radio. En ellas se transmite información aprovechando la naturaleza de la propagación de las ondas electromagnéticas, modulando una portadora en amplitud (AM) o frecuencia (FM) mediante técnicas puramente analógicas. La variación de la corriente de campo de un motor de continua para el control de su velocidad también es una señal que podemos denominar analógica.

SISTEMAS DIGITALES

Los sistemas electrónicos nunca son totalmente analógicos, pero tampoco estrictamente digitales; son híbridos. En este apartado, vamos a profundizar en los sistemas digitales.

El término digital proviene de **dígito**, sinónimo de dedo, y nos acerca al mundo de lo discreto, de lo que podemos contar; en definitiva, de lo **discontinuo**. En cierto sentido, no necesitamos los números reales para cuantificar un fenómeno, sino que nos alcanza con los números enteros. Debemos destacar que el hecho de que un sistema sea digital no implica que se trabaje estrictamente con números binarios. Un sistema digital puede tranquilamente ser de naturaleza decimal.

Lo que sucede es que **el sistema de numeración binario** se presta de manera excepcional para brindar soluciones a infinidad de cuestiones vinculadas a la ingeniería electrónica: desde lo estructural, pasando por lo matemático, hasta las ventajas logra-

En electrónica, todo sistema trabaja con señales digitales de naturaleza binaria

das en el almacenamiento, procesado, fiabilidad y transporte de información. Es por eso que el sistema de numeración binario se vuelve indispensable en el diseño, los dispositivos conversores A/D (analógico-digitales) y los conversores D/A (digitales/analógicos), ver **Figura 1**.

En electrónica, todo sistema trabaja con señales digitales de naturaleza binaria. Esto implica la presencia de solo dos estados posibles: conducción (ON) y corte (OFF), en equivalencia a verdadero y falso, nociones que maneja todo sistema lógico.

Los transistores de los circuitos integrados actúan como llaves de conmutación, al permitir la conducción o interrupción de un circuito eléctrico modificando el estado del sistema. Es decir, gobiernan el comportamiento lógico del circuito. Hablaremos entonces de transistores y tecnologías que se ajustarán a niveles de tensión representativos de dichos estados, ver **Tabla 1**.

VALORES	ESTADOS
5 V / 3,3 V / 1,8 V	ON (1 en binario o estado alto)
0 (cero) V	OFF (0 en binario o estado bajo)

TABLA 1. **Todas las operaciones lógicas podrán efectuarse en estas condiciones.**



LÓGICA COMBINACIONAL Y LÓGICA SECUENCIAL

En este punto, debemos hacer una distinción entre lógica combinacional y lógica secuencial. La **combinacional** se refiere a un sistema que reacciona siempre de la misma manera frente al mismo juego o combinación de entradas. Es decir, cada vez que se aplica una combinación de entradas determinada, el sistema devuelve el juego de salidas correspondiente. Esta es una operación sin memoria.

Un sistema **secuencial,** en cambio, tiene memoria. Responde no solo a una determinada combinación de entradas, sino que también coteja con algún resultado anterior para realizar una acción.

Es decir, depende del orden y de la secuencia con que se ejecutan las combinaciones a su entrada. Es importante aclarar que todo circuito digital puede desglosarse en dos grandes bloques: uno **combinacional** y otro **secuencial**, que actúan en conjunto (**Figura 2**).

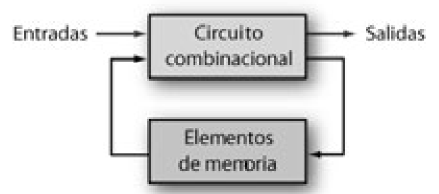


FIGURA 2. Vemos aquí la conformación de un sistema digital que integra módulos.

LOS SISTEMAS DE NUMERACIÓN

La necesidad de representar cantidades de un determinado objeto mediante símbolos ha llevado a desarrollar diversos métodos de representación. Analizaremos a continuación los sistemas de numeración actuales. Históricamente, los esfuerzos se centraron en encontrar un sistema que precisara de la menor cantidad de símbolos para representar grandes cantidades, y que facilitara las operaciones y cálculos. Los sistemas posicionales de numeración surgieron en forma independiente, tanto en Oriente como en América. En ambos casos, hay un símbolo representante de la **ausencia de cantidad: el cero**.

Los sistemas posicionales utilizan un conjunto limitado y constante de símbolos, donde cada uno representa una cierta cantidad de unidades. Pero, además, dependiendo de la posición que ocupe en el grupo de caracteres de representación, este símbolo tendrá mayor o menor peso. Nuestro sistema decimal, por ejemplo, es un ejemplo típico de un sistema de **representación posicional**. Lo llamamos decimal pues, con la combinación de 10 dígitos, es posible representar cualquier cantidad: 0,1, 2, 3, 4, 5, 6, 7, 8 y 9. Veamos un ejemplo del peso de los símbolos de acuerdo con su posición. Podemos, entonces, descomponer la cantidad **18.127** de la siguiente forma:

18.127 = 1 x 10.000 + 8 x 1.000 + 1 x 100 + 2 x 10 + 7

El decimal es un sistema de representación posicional de base 10; el binario es de base 2 y el hexadecimal, de base 16



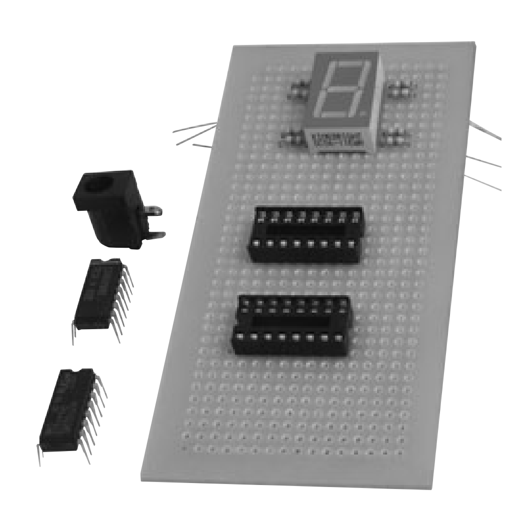
REGLA POSICIONAL

Para todo sistema entero de numeración posicional se cumple que el último dígito de la derecha representa unidades (**peso de valor 1**). El peso de cada posición se incrementa de derecha a izquierda en potencias de la base.

Observemos que cada dígito que conforma el número **18127** tiene un peso propio por la posición que ocupa en la cadena de caracteres. El peso de cada dígito en el sistema decimal es claramente múltiplo de **10**. Luego, la descomposición que sigue también puede lograrse:

$$\mathbf{18127 = 1 \times 1^{04} + 8 \times 1^{03} + 1 \times 1^{02} + 2 \times 1^{01} + 7 \times 1^{00}}$$

De aquí viene la denominación de sistema en **base 10**, sinónimo de sistema decimal. Este tipo de descomposición puede extenderse a los demás sistemas de numeración, como el binario que desarrollaremos más adelante.



Sistema **binario**

Estudiaremos en este apartado el sistema binario de numeración, utilizado en los sistemas digitales como base de operación matemática, almacenamiento y representación.

Se denomina sistema binario al sistema de numeración de base 2, que, como se desprende, consta de solo 2 dígitos para representar cantidades: 0 y 1. Con ellos podemos representar **22 = 4** valores; con 3 **dígitos, 23 = 8** valores; con 4 dígitos, **24 = 16** valores y así sucesivamente; por lo tanto, con N dígitos se podrá representar hasta **2N** valores. Es importante destacar que los ceros a la izquierda no cuentan, como es lógico, a la manera en que estamos acostumbrados en el sistema decimal. Para operar con números binarios, lo haremos intuitivamente del mismo modo que cuando utilizamos el sistema decimal, ver **Tabla 2**.

2 BITS	3 BITS	4 BITS	VALOR DECIMAL
00	000	000	0
01	001	0001	1
10	010	0010	2
11	011	0011	3
	100	0100	4
	101	0101	5
	110	0110	6
	111	0111	7
		1000	8
		1001	9
		1010	10
		1011	11
		1100	12
		1101	13
		1110	14
		1111	15

TABLA 2. Los números binarios pueden formarse a partir de arreglos de distintas cantidad de dígitos.

OPERACIONES CON NÚMEROS BINARIOS

En las operaciones típicas de suma, resta y multiplicación, se aplican las técnicas de acarreo, adaptadas en este caso a la operación con solo 2 símbolos (**Figura 3**). Esto es: **01 + 01**, en binario, arrojará el valor 10, ya que **1 + 1** no puede representarse con un solo símbolo. Se deja entonces un 0 en la posición menos significativa y se acarrea un 1 hacia la más significativa. El resultado es 10 binario (2 en decimal).

La multiplicación es todavía más intuitiva y se realiza de manera habitual. La única posibilidad de que una multiplicación entre 2 bits arroje 1 como resultado es que ambos sean 1. Es importante destacar que agregar un cero por derecha a un número binario tiene como resultado duplicar su valor. En efecto, dado el número 110 binario (6 decimal), el número 1100 (binario) corresponde al doble de su valor (12 en decimal).

REPRESENTACIÓN CON SIGNO

Es posible representar números binarios signados utilizando el bit más significativo como bit de signo: un 1 indicará negativo y un 0, positivo. Por ejemplo, como se ve en la **Tabla 3**, el número **1010** (10 decimal en binario **no signado**) representaría el número **-6 decimal en binario signado**.

Debido a que un bit se utiliza como signo, el número máximo que es posible representar para los positivos es **0111** (**7 decimal**). El mínimo negativo será el **1000** (**-8 decimal**), y el máximo negativo, el **1111** (**-1 decimal**).

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

TABLA 3. Representación de números binarios con signo.

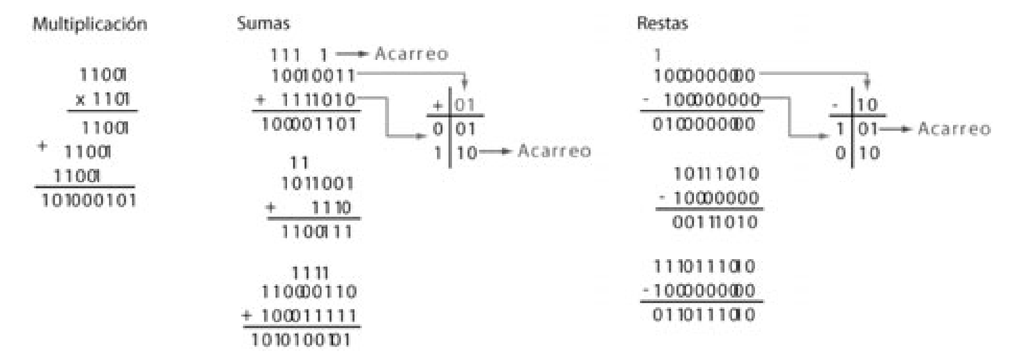


FIGURA 3. Observamos aquí las operaciones de suma, resta y multiplicación entre números binarios.

CONCEPTO DE MÓDULO

En representaciones de números binarios signados, se llama módulo al resultado de sumar la representación negativa de un número con su representación positiva. Por ejemplo, si sumamos **1111** (**-1 decimal**) a **0001** (**1 decimal**), obtendremos el módulo: **10000**. Se dice entonces que estos números son complementarios con respecto a su módulo.

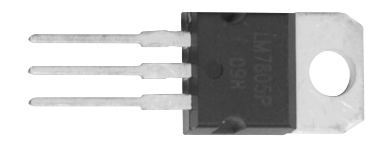
TRABAJO CON BINARIOS SIGNADOS

Cuando trabajamos con binarios signados, calculamos el módulo como el número binario correspondiente a la cantidad de combinaciones que se pueden lograr con la cantidad de bits utilizada incluyendo el signo. Por ejemplo, para el caso de utilizar 4 bits (incluido el signo), se puede lograr **24 = 16** combinaciones (de -8 a 7 pasando por el cero). Por lo tanto, el módulo de este arreglo de bits signados es 16; en binario: 10000.

Conociendo el módulo, es posible determinar la representación de números negativos, simplemente operando sobre la representación positiva, sin necesidad de tener a la vista todas las combinaciones posibles. Restando entonces al módulo la representación positiva, obtendremos la representación negativa de ese número con esa cantidad de bits (**Figura 4**).

```
  10000   (Módulo 24 = 16 en decimal)
-  0110   (Representación positiva: 6 decimal)
  -----
   1010   (Representación negativa: -6 decimal)
```

FIGURA 4. **Cálculo de la representación negativa del número** 0110 (6 decimal). **Para módulo** 10000 (24 = 16 decimal).



Se llama módulo al resultado de sumar la representación negativa de un número con su representación positiva

¿Por qué conocer **el sistema binario?**

Cuando hablamos de electrónica digital, nos referimos a sistemas electrónicos que procesan, almacenan, se comunican y operan en binario. Veremos aquí su importancia.

Un sistema digital de este tipo manejará internamente solo dos estados: 1 (alto) y 0 (bajo). En consecuencia, la organización de la información estará basada en arreglos de valores binarios. Surge entonces el concepto de **bit**, que no es más que el acrónimo de **binary digit** (dígito binario). Con un bit podremos representar dos estados o valores. El arreglo de 8 bits se denomina **byte**, término que deriva de la palabra anglosajona *bite*, "**mordisco**" en castellano. Se refiere a

la cantidad mínima de datos que un procesador puede "morder" a la vez, adoptándose por convención el tamaño de 8 bits. La traducción al español que toma la Real Academia es **octeto**, pero nosotros utilizaremos **byte** para evitar confusiones. De allí, las unidades de capacidad y almacenamiento más utilizadas:

- **Kilobyte = 1024 bytes**
- **Megabyte = 1024 Kb**
- **Gigabyte = 1024 MB**
- **Terabyte = 1024 GB**

Con un byte podemos manejar 256 valores posibles. Por ejemplo, en programación, se acostumbra definir variables de tipo carácter, de 1 byte de longitud. Se trabaja, así, con valores enteros en un rango de 0 a 255 (sin signo) o de -128 a 127 (con signo).
Es común también hablar de "words" o arreglos de 16 bits, no solo en programación de sistemas con microprocesadores y microcontroladores, sino también en algunos medios de almacenamiento, como memorias RAM y ROM, que proponen direccionamiento y palabras de datos mínimas de 16 bits. Por lo tanto, es una unidad de trabajo estándar que identifica una palabra de 2 bytes de longitud, muy utilizada para definir variables de tipo entero, capaces de manejar 65.536 valores distintos, suficientes para muchas de las operaciones más corrientes.

El sistema hexadecimal consta de 16 símbolos para representar números

DECIMAL	BINARIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

TABLA 4. En la tabla se muestra una equivalencia entre valores decimales, binarios y hexadecimales.



ALCANCE HEXADECIMAL

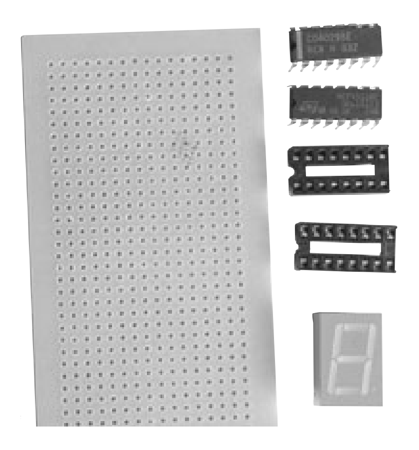
El sistema hexadecimal consta de **16** símbolos para representar números. Del **0** al **9** coinciden con los símbolos del sistema decimal. Luego, se agregan los caracteres **A**, **B**, **C**, **D**, **E** y **F** para obtener el juego de 16 símbolos, recordemos que **162 = 256**.

CONVERSIÓN DECIMAL A BINARIO

Para convertir un número de decimal a binario, hay que dividir el decimal sucesivamente por **2**, hasta obtener un cociente menor que el divisor. Este cociente (que será **0** o **1**, naturalmente), más los restos de las sucesivas divisiones efectuadas, constituye la representación binaria buscada. De esta forma, el cociente de la división será el dígito más significativo del número binario, y el menos significativo corresponderá al primer resto de la división (**Figura 5**).

```
18 |_2
 0   9 |_2
     1   4 |_2
         0   2 |_2
             0   1   = 10010b
```

FIGURA 5. Se muestra aquí el método de divisiones sucesivas para hallar la representación binaria de un número decimal.



CONVERSIÓN DECIMAL A HEXADECIMAL

Este método puede extenderse a conversiones de decimal a cualquier otro tipo de base, por ejemplo, hexadecimal. Para esto, habrá que dividir el número decimal sucesivamente por la base en la que se lo quiere representar; en nuestro caso, **16** (**Figura 6**).

```
5680 |_16
 88    355 |_16
  80    35    22 |_16
   0     3     6     1     = 1630h
```

FIGURA 6. El mismo método de divisiones sucesivas para hallar la representación ahora hexadecimal de un número decimal.

El orden de los dígitos del número hexadecimal obtenido es el inverso del que hemos obtenido en la división anterior.

CONVERSIÓN BINARIO A DECIMAL

Para realizar este procedimiento, hacemos el desarrollo en potencias de 2 de un número binario y sumamos los pesos:

$$\mathbf{110101b = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 53d}$$

Es decir, en orden ascendente, el peso de cada dígito binario queda determinado: 1, 2, 4, 8, 16, 32, 64 (etcétera). Observamos cómo aumenta en potencias de 2, duplicando el peso en cada dígito. La conversión a decimal se realiza efectuando la suma de las contribuciones de peso de cada dígito que tome valor 1. En este caso, el número es 53.

CONVERSIONES DIRECTAS ENTRE BINARIO Y HEXADECIMAL

En los pasajes de binario a hexadecimal, y viceversa, se da una situación particular. Como una base es potencia de la otra, en nuestro caso **16 = 24**, es posible demostrar que los dígitos de la base menor (base 2 en nuestro caso) pueden agruparse en un número igual al exponente (4 en el ejemplo) para conformar un dígito de la base mayor.

BINARIO	CUARTETOS	HEXADECIMAL
1110100101	(0011) (1010) (0101)	3A5h
101111	(0000) (0010) (1111)	02Fh
101011100001	(1010) (1110) (0001)	AE1h

TABLA 5. Ejemplo de conversiones de binario a hexadecimal.

Para simplificar: partiendo un número binario en cuartetos (agregando ceros a la izquierda a la parte más significativa, si es necesario), el reemplazo de dicho cuarteto por el símbolo hexadecimal equivalente se realiza en forma directa (ver **Tabla 5**).

CONVERSIÓN INVERSA

La conversión inversa, de hexadecimal a binario, se lleva a cabo en forma directa y de la misma manera. Se reemplaza cada símbolo hexadecimal por un número binario de 4 bits. Por lo tanto, no se requiere ningún tipo de operación para los procedimientos de este tipo. Es por eso que el sistema hexadecimal es tan utilizado en sistemas electrónicos e informáticos. Para las personas resulta difícil trabajar en binario debido a que es fácil perderse con largas secuencias de unos y ceros para operar. El sistema hexadecimal permite representar de manera más compacta la información contenida en largas secuencias binarias que se almacenan y procesan en los equipos digitales.

El sistema hexadecimal es muy utilizado en sistemas electrónicos e informáticos

CÓDIGO BCD

Este tipo de codificación está diseñado con el objetivo de simplificar la conversión de decimal a binario, y evita la necesidad de recurrir a tediosas operaciones aritméticas, como en el método de divisiones sucesivas por 2.



SUMA DE BCD

Para sumar números codificados en **BCD**, simplemente operamos como si trabajáramos con números naturales. Si la suma parcial de un cuarteto supera el valor **1001** (**9** en decimal), se suma al resultado **0110** (**6**) y se acarrea **0001** al siguiente dígito BCD o cuarteto de bits.

De este modo, el **código BCD** no es más que la representación binaria con 4 bits de cada dígito de un número decimal, en forma individual. Las conversiones de BCD a decimal y de decimal a BCD se realizarán, entonces, directamente, por simple inspección: **9723d = (1001) (0111) (0010) (0011) BCD.** Sin embargo, este tipo de representación es una equivalencia; no es una conversión matemática como el traspaso de binario a hexadecimal o de hexadecimal a binario. No estamos representando números en ninguna base. Por lo tanto, la suma de los pesos de los bits individuales no arrojará el valor decimal representado en BCD (**Figura 7**).

```
        111
        0111    1001      79
      + 0011    0101    + 35
        11 1    11       114
        1010    1110
        0110    0110
0001    0001    0101
  1       1       4
```

FIGURA 7. Vemos aquí la operación de suma de números BCD.

Las compuertas **lógicas**

Los circuitos lógicos digitales han revolucionado la electrónica, ya que son mucho más inmunes al ruido eléctrico, más rápidos y más versátiles que su contraparte analógica.

El continuo avance de la tecnología permite la aplicación de la electrónica digital en cada vez más áreas de especialización. Aquí veremos los bloques constitutivos de los circuitos digitales: las **compuertas lógicas**.

Un circuito lógico puede representarse con un mínimo de detalle, como una caja negra con una determinada cantidad de entradas y salidas. Entendemos por caja negra a un circuito del cual no tenemos datos sobre su composición interna, pero sobre el que sí podemos conocer cómo es su salida ante una cierta entrada. Como las entradas al circuito lógico pueden tomar solo los valores discretos **0** y **1**, la lógica llevada a cabo por él puede describirse completamente a través de una tabla que ignora su comportamiento eléctrico y define la salida con **0** y **1** (**Figura 8**).

VALORES LÓGICOS

Los valores lógicos se representan con un **0** o un **1**, y se denominan **dígitos binarios** o bits. Cada uno representa un rango de validez para una señal analógica. Un circuito lógico, cuya salida depende solo

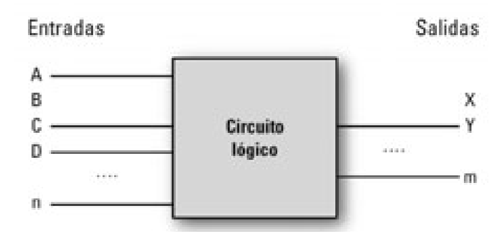


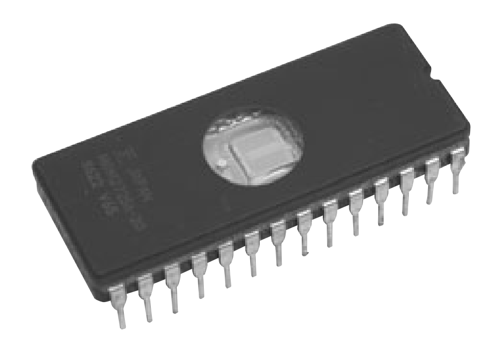
FIGURA 8. Representación de un circuito lógico. Las n **entradas aplicadas producen las** m **salidas. Desde este punto de vista, no resulta necesario conocer su estructura interna.**

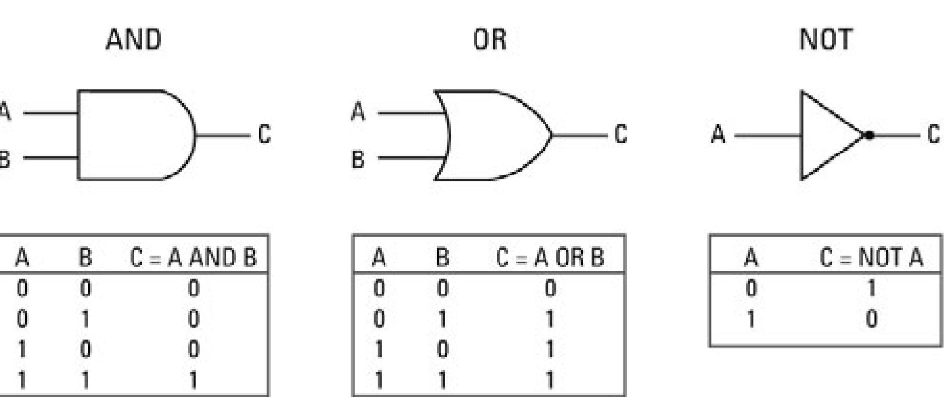
de sus entradas presentes, se conoce como **circuito combinacional,** y su funcionamiento se representa por medio de una **tabla de verdad**.

Cualquier circuito combinacional puede construirse sobre la base de tres compuertas lógicas fundamentales, denominadas **AND**, **OR** y **NOT**. Existe otra compuerta particular llamada **BUFFER**, en donde la señal lógica no sufre ningún cambio, es decir que la tensión de salida sigue a la de entrada. Estas compuertas se utilizan generalmente para regenerar señales débiles y convertirlas otra vez en señales fuertes para que puedan ser transmitidas a lo largo de cierta distancia sin pérdida de información.

En la **Figura 9** observamos la representación gráfica de la compuerta **NOT.** En la punta del triángulo se ha colocado un círculo que denota el carácter inversor de la función, e implica que el valor lógico presente en la entrada de la compuerta se invierte a la salida. De hecho, a la compuerta **NOT** también se la conoce con el nombre **inversor**. Si la entrada al inversor es **A**, entonces este implementa la función **NOT A** y se representa con **$\bar{A}$** (nombre de la entrada con una raya en la parte superior).

¿Qué sucede si en cada entrada de la función **OR** colocamos un inversor? La configuración lógica que se obtiene se conoce con el nombre **NAND** y es la implementación inversa de la función **AND** (**Figura 10**).



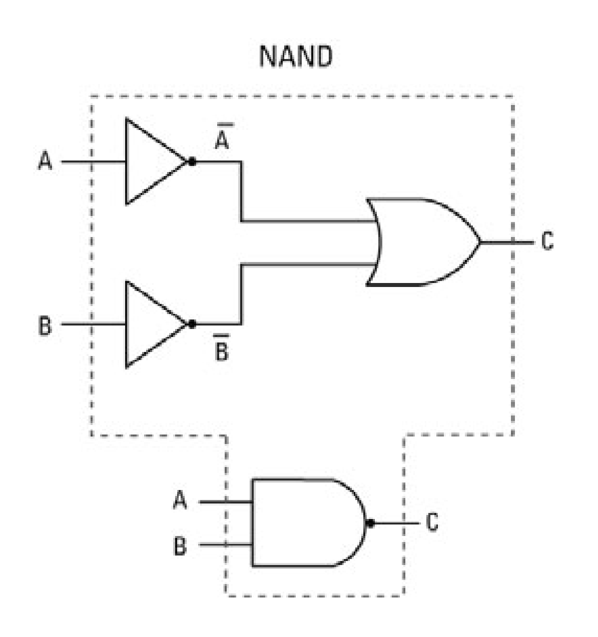


A	B	C = A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	C = A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	C = NOT A
0	1
1	0

FIGURA 9. Compuertas lógicas elementales AND, OR **y** NOT **y compuerta BUFFER. Su representación gráfica y su tabla de verdad, a partir de las cuales es posible construir cualquier circuito combinacional.**



A	B	C = A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

FIGURA 10. Compuerta lógica NAND. Implementación como compuerta OR con sus entradas negadas y su tabla de verdad. La compuerta NAND produce el resultado inverso de la compuerta AND.

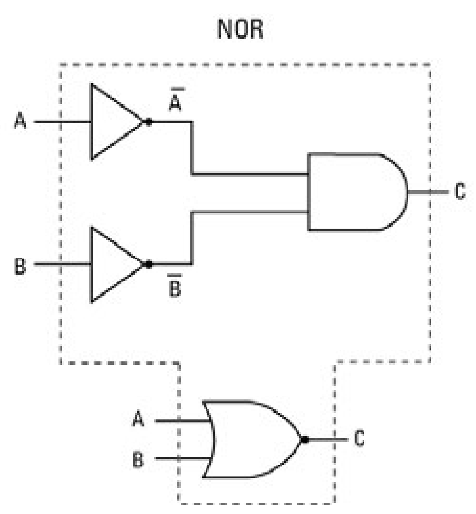
Ahora, veamos qué obtenemos si usamos la compuerta **AND** y colocamos inversores en sus entradas. La configuración lógica resultante se conoce como **NOR** y es la implementación inversa de la función **OR** (**Figura 11**).

Observemos la configuración lógica que se muestra en la **Figura 12**. Ésta se conoce como **OR-Exclusiva** o **XOR**, y su característica es que produce una salida lógica **1** cuando sus entradas son diferentes, mientras que arroja una salida lógica **0** cuando sus entradas son iguales.

DE LA ELECTRÓNICA A LA LÓGICA

Dijimos que un circuito lógico puede representarse como una caja negra con entradas y salidas. Esta caja negra es la que contiene un circuito electrónico que implementa la compuerta lógica representada por el bloque combinacional. En las páginas siguientes veremos algunos circuitos electrónicos muy básicos que implementan las compuertas lógicas elementales vistas anteriormente





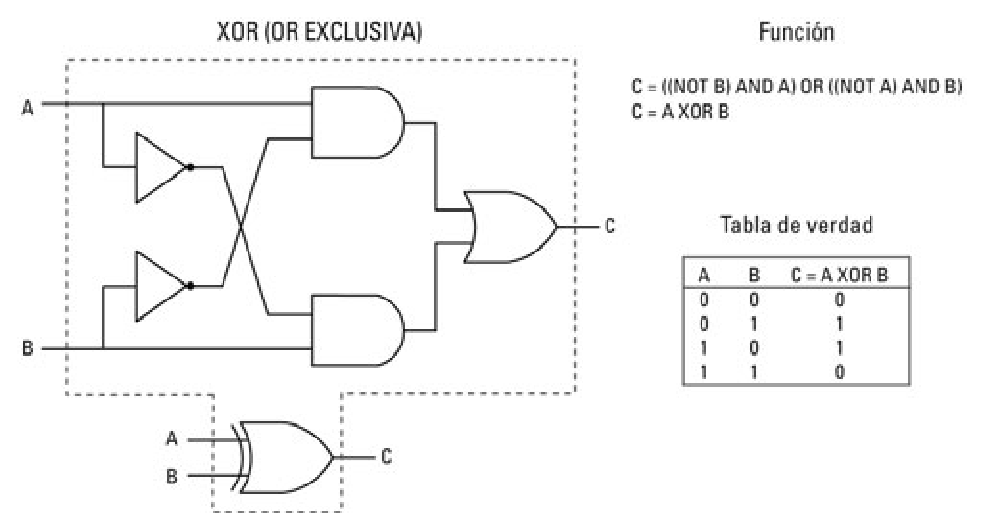
Función

C = (NOT A) OR (NOT B)
C = A NOR B

Tabla de verdad

A	B	C = A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

FIGURA 11. **Compuerta lógica NOR. Implementación como compuerta AND con sus entradas negadas y su tabla de verdad. La compuerta NOR produce el resultado inverso de la compuerta OR.**



A	B	C = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

FIGURA 12. **Compuerta lógica XOR. Implementación como secuencia AND-OR y su tabla de verdad. Se utiliza para identificar cuando dos entradas son iguales o distintas entre sí.**

En el circuito de la **Figura 13**, si en cualquier entrada (**A**, **B**) se presenta un valor de tensión positiva que haga conducir al diodo, este valor se observa a la salida (**S**). En caso contrario, el resistor fuerza 0 V. Entonces, dado que cualquier entrada que esté en **1** ocasiona un **1** a la salida, esta es una compuerta **OR**, pues esta característica se corresponde con la tabla de verdad de dicha compuerta.

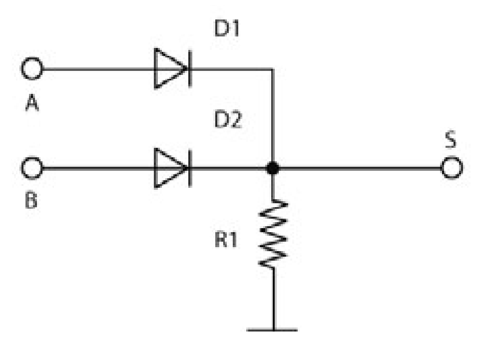


FIGURA 13. Una compuerta OR elemental con diodos y resistores (RDL o Resistor Diode Logic).

En el circuito de la **Figura 14**, si cualquier entrada (**A**, **B**) se conecta a 0 V, este valor se observa a la salida (**S**). En caso contrario, el resistor fuerza la tensión de alimentación. Entonces, dado que cualquier entrada que esté en **0** ocasiona un **0** a la salida, esta es una compuerta **AND**, pues esta característica se corresponde con la tabla de verdad de dicha compuerta.

Las compuertas lógicas implementan las funciones lógicas elementales

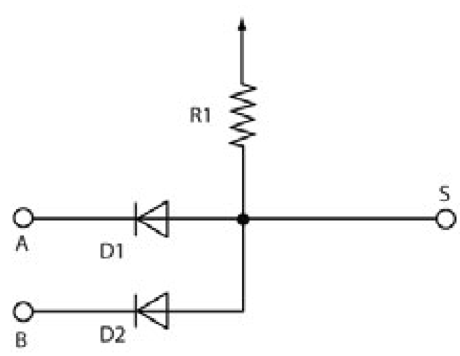


FIGURA 14. Una compuerta AND **elemental con diodos y resistores (**RDL **o Resistor Diode Logic).**

El circuito de la **Figura 15** utiliza un transistor tipo **NPN**, **Q1,** para implementar la compuerta **NOT**. En este caso, con una tensión de **1** lógico en el punto **A,** el transistor entra en saturación, y el punto **C** se coloca a la tensión de **0** lógico. Cuando el transistor entra en corte a través de la aplicación de una tensión de **0** lógico en el punto **A**, el punto **C** queda a tensión de fuente menos la caída de tensión en el resistor **R1**.

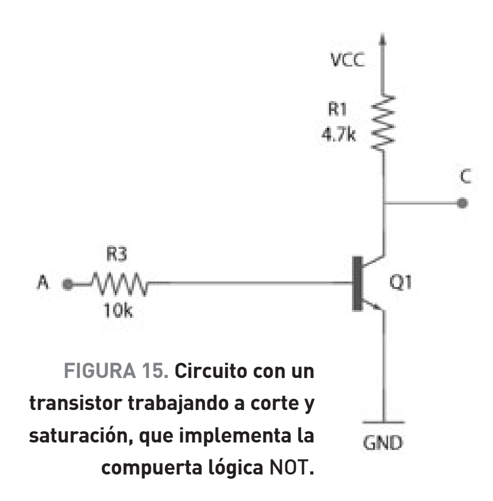


FIGURA 15. Circuito con un transistor trabajando a corte y saturación, que implementa la compuerta lógica NOT.

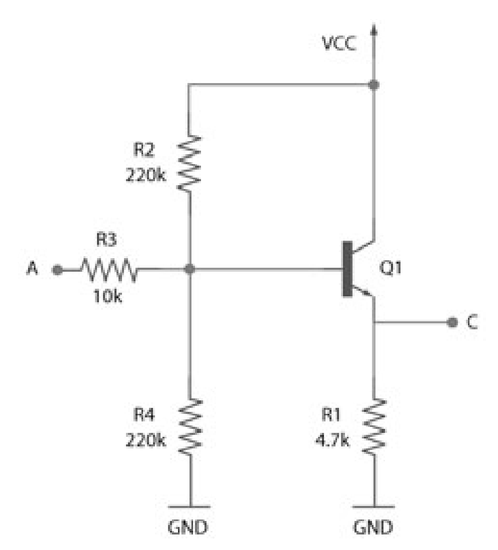


FIGURA 16. Circuito con un transistor, que implementa la compuerta BUFFER.

En la **Figura 16** observamos un circuito que implementa la compuerta BUFFER o seguidor de tensión. Cuando en el punto **A** tenemos una tensión correspondiente a un **0** lógico, el transistor **Q1** está en corte, y en el punto **C** tendremos una tensión equivalente a un **0** lógico. Cuando en **A** tenemos un **1** lógico, **Q1** conduce y coloca en el punto **C** a la tensión de fuente o **1** lógico.

En la **Figura 17** se presenta la implementación de la compuerta **NOR** agregando un inversor (basado en un transistor PNP) en cascada con la compuerta **OR** que ya vimos en la **Figura 13**.

La **Figura 18** muestra la implementación de la compuerta **NAND** agregando un inversor (esta vez, basado en un transistor PNP) en cascada con la compuerta **AND** vista en la **Figura 14**.

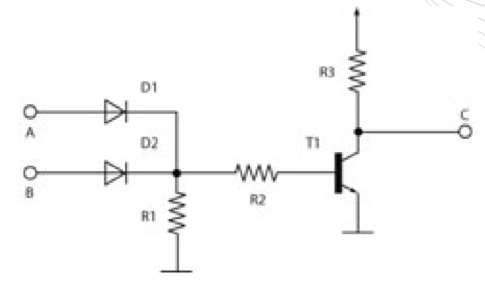


FIGURA 17. Circuito elemental que implementa la compuerta lógica NOR.

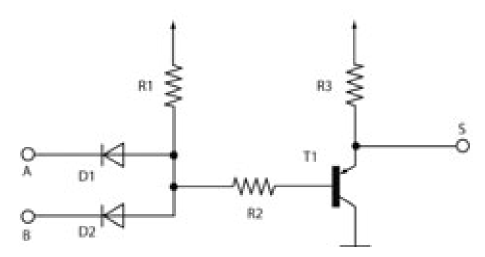


FIGURA 18. Circuito elemental que implementa la compuerta lógica NAND.

BUFFERS DE TRES ESTADOS

El diseño electrónico de las salidas de algunos dispositivos **CMOS** o **TTL** puede estar en un estado lógico **0**, **1** o de **alta impedancia,** también llamado **Hi-Z**. En la representación gráfica de un BUFFER de tres estados (**Figura 19**), se distinguen con claridad las entradas y las salidas. Adicionalmente, se representa una señal que actúa sobre el BUFFER, denominada **habilitación de salida,** que puede ser activo alto o bajo, dependiendo de si está presente o no el círculo que denota inversión. Cuando esta entrada está activa, el dispositivo se comporta como un BUFFER normal, mientras que si está negada, entonces la salida del BUFFER entra en un estado de alta impedancia y, funcionalmente, se comporta como si no estuviera allí.

La utilidad de estas compuertas es que permiten a múltiples fuentes compartir una sola línea de comunicación, mientras que solo una de ellas transmite datos por vez. Es decir, cuando un dispositivo quiere colocar información en la línea, deberá salir de su estado **Hi-Z** y empezar la transmisión, pero antes de hacerlo, debemos asegurarnos de que los demás dispositivos en la línea ingresaron en su estado **Hi-Z**; de lo contrario, habrá colisión de datos.

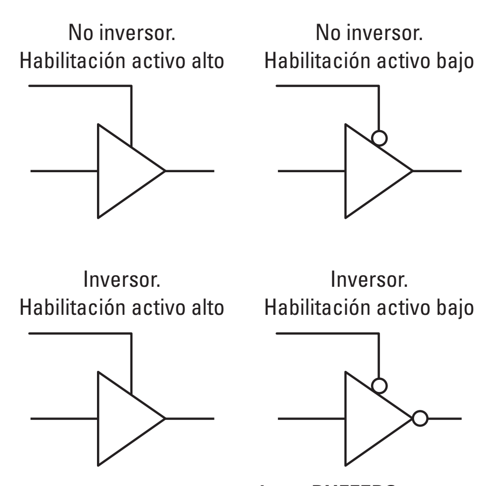


FIGURA 19. Representación de BUFFERS de tres estados, inversores y no inversores, cada uno con su respectiva habilitación activo alto o bajo.

COMPUERTAS A COLECTOR ABIERTO

La salida de compuertas **TTL** a colector abierto es otra de las configuraciones que pueden tener los dispositivos pertenecientes a esta familia. Se logra a través de una modificación interna de la compuerta TTL básica, que permite poner el colector del transistor de salida al alcance del usuario. Es decir, la salida de la compuerta es el colector de uno de los transistores con los que esta se encuentra construida.

De esta manera, entonces, para lograr el correcto funcionamiento de la salida, es necesario colocar externamente un resistor de carga. Este requiere de un punto de referencia de tensión, que no necesariamente tiene que coincidir con la alimentación de la compuerta. El principal objetivo que se busca en este tipo de configuraciones es obtener un mayor nivel de corriente que pueda manejar la compuerta (**Figura 20**).

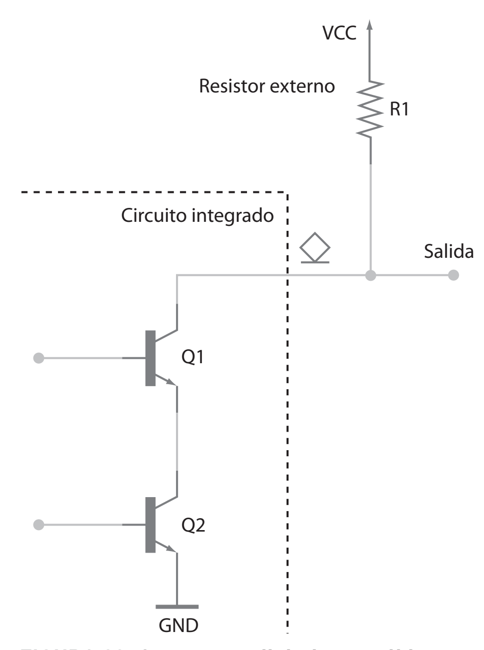


FIGURA 20. Compuerta digital con salida a colector abierto. Desde el exterior del circuito integrado es posible acceder al colector del transistor de salida. Cabe observar la notación para compuertas con salidas de este tipo: un rombo con una raya horizontal en su parte inferior.

Asimismo, y como el resistor externo puede estar conectado a un punto de tensión diferente de la alimentación de la compuerta, este tipo de dispositivos es ampliamente utilizado en aplicaciones en las que se necesita vincular dos familias lógicas que tienen umbrales diferentes (**CMOS** y **TTL**, por ejemplo).

También, como es posible manejar mayor potencia en forma directa desde la compuerta, se puede usar estos dispositivos para controlar pequeñas cargas, tales como lámparas, LEDs y relays. Otra aplicación de las compuertas a colector abierto es en lógica cableada, en donde la salida de varias compuertas se conecta a un resistor externo conectado a la tensión de fuente.

RESISTORES DE PULL-UP

Los resistores de pull-up se utilizan en circuitos lógicos digitales y se colocan en las entradas de los dispositivos lógicos. Su misión es asegurar que dichas entradas mantengan siempre un nivel lógico correcto y definido, para evitar que la entrada quede flotando. Una entrada flotante provoca un inadecuado funcionamiento de la compuerta y ofrece a la entrada propiamente dicha un nivel de tensión indefinido. Los resistores de **pull-up** elevan la tensión de la entrada donde están conectados a un determinado nivel, que suele ser la tensión de fuente (**Figura 21**).

El resistor propiamente dicho, sin embargo, debe tener un valor que haga débil la línea, en el sentido de

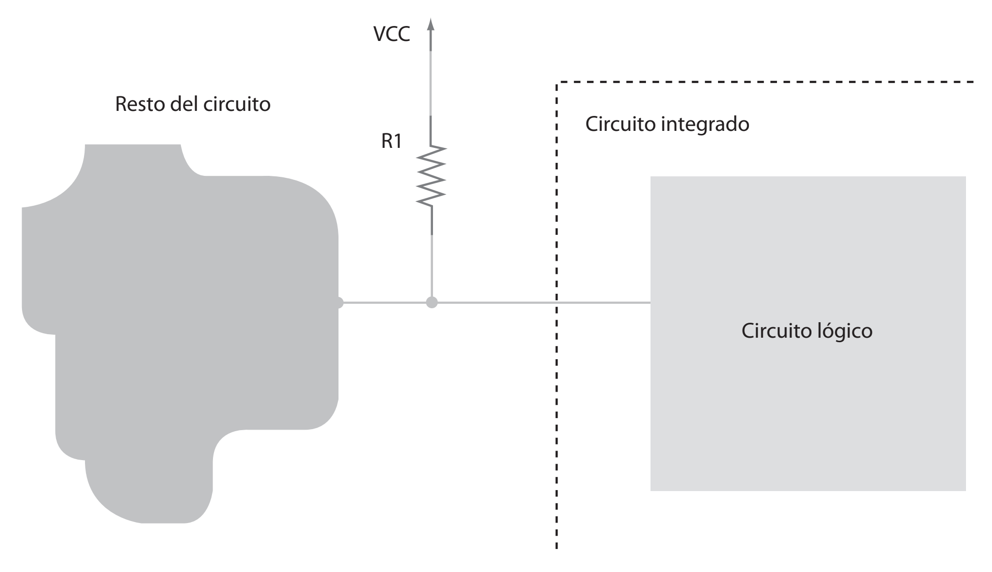


FIGURA 21. En esta figura podemos ver cómo los resistores de pull-up se colocan en las entradas para garantizar un nivel lógico definido en ellas.

que si otro dispositivo trata de imponer un nivel de tensión distinto en ella, el pull-up no se va a resistir y cederá sin inconvenientes.

La función principal de los resistores de pull-up es prevenir un exceso de corriente en el circuito, que ocurriría si un dispositivo tratara de llevar un punto a un determinado nivel de tensión cuando este ya tuviera uno distinto.

Así como existen los resistores de pull-up, también están los de **pull-down**, que son idénticos a los primeros, excepto que en vez de elevar la tensión de una entrada lógica a cierto nivel de tensión, la bajan a nivel de tierra o masa.

Los resistores de pull-up generalmente consumen menos potencia que los de pull-down. Por este motivo, son preferidos en los circuitos digitales donde la potencia consumida suele ser un tema crítico a la hora de diseñar.

LÓGICA CABLEADA

Se conoce con este nombre a las conexiones que implementan compuertas lógicas mediante la conexión directa de dispositivos de colector abierto o equivalente (drenaje abierto en **MOS**). Cuando cualquiera

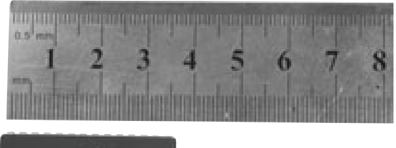


de los transistores está en conducción, el bus se encuentra a nivel lógico **0**. Solo cuando todos estén al corte, obtendremos la tensión de la fuente de alimentación a través del resistor **R1**. Dependiendo de la lógica que activa al transistor de colector abierto, activo alto o activo bajo, esta conexión se conoce como **wired-AND** (**AND cableada**) o **wired-OR** (**OR cableada**), respectivamente.

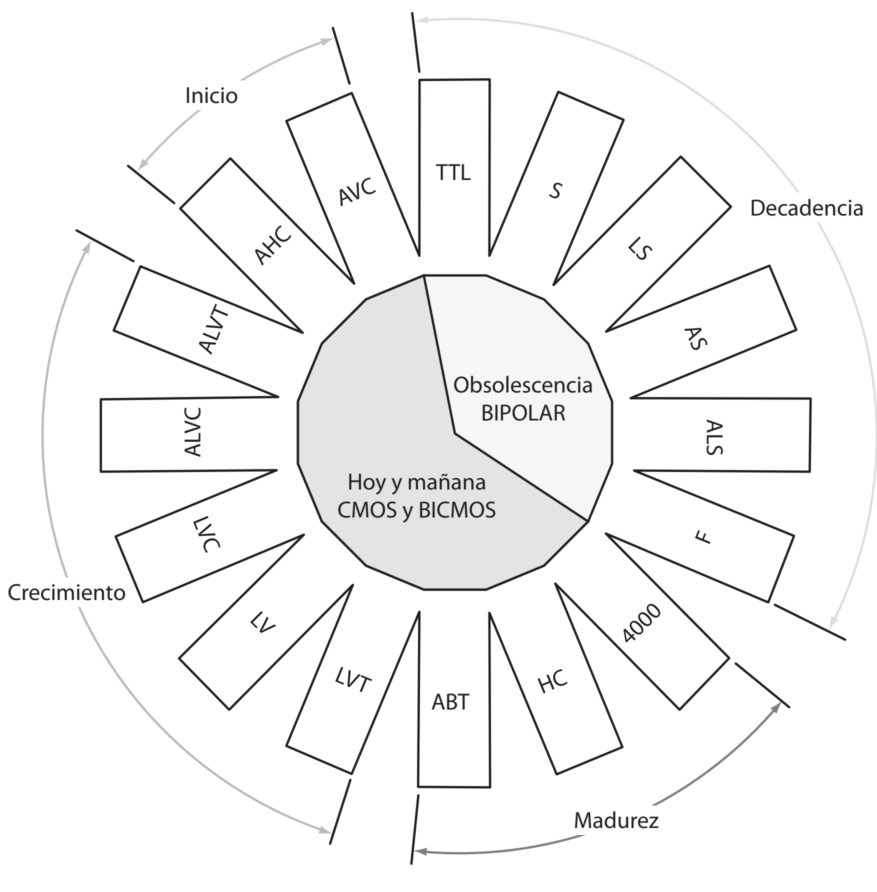
FAMILIAS LÓGICAS, NIVELES Y UMBRALES

Con el objetivo de lograr mejores prestaciones en los circuitos lógicos digitales, se viene dando una constante evolución que da origen a las distintas **familias lógicas** (**Figura 22**). Dentro de ellas, hay diversas subfamilias con características distintivas. Esta evolución que va experimentando cada una de las tecnologías pasa por varias etapas: **inicio** y **crecimiento**, **madurez** y **decadencia** o desuso. Pero siempre el objetivo buscado por cualquiera de estas tecnologías es **reducir el consumo** e **incrementar la velocidad de conmutación**.

Los resistores de pull-up se colocan en las entradas de dispositivos lógicos para forzar el estado alto







TTL	Transistor-Transistor Logic
S	TTL Schottky
LS	TTL Schottky bajo consumo
AS	TTL Schottky mejorada
ALS	Versión mejorada de LS
F	TTL de alta velocidad
4000	Serie CMOS
HC	CMOS de alta velocidad
ABT	Tecnología BICMOS avanzada
LVT	BICMOS de baja tensión
LV	Baja tensión
LVC	CMOS de baja tensión
ALVC	CMOS de baja tensión mejorada
ALVT	BICMOS de baja tensión mejorada
AHC	CMOS de alta tensión mejorada
AVC	CMOS de muy baja tensión mejorada

FIGURA 22. Representación de familias lógicas en etapas de inicio, crecimiento, madurez y decadencia. Vale observar que la tecnología bipolar está pasando su etapa de madurez y tiende al desuso.

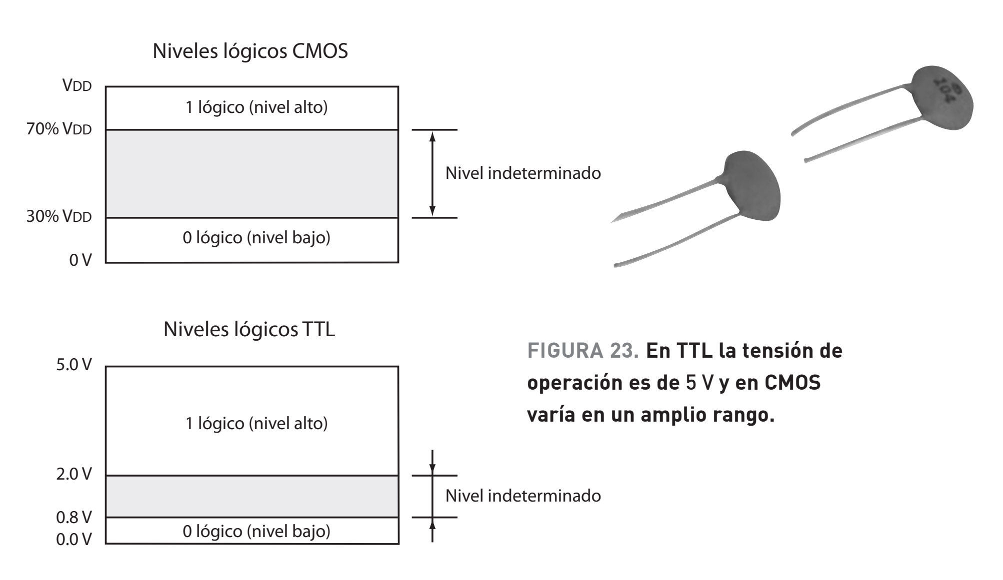


FIGURA 23. **En TTL la tensión de operación es de 5 V y en CMOS varía en un amplio rango.**

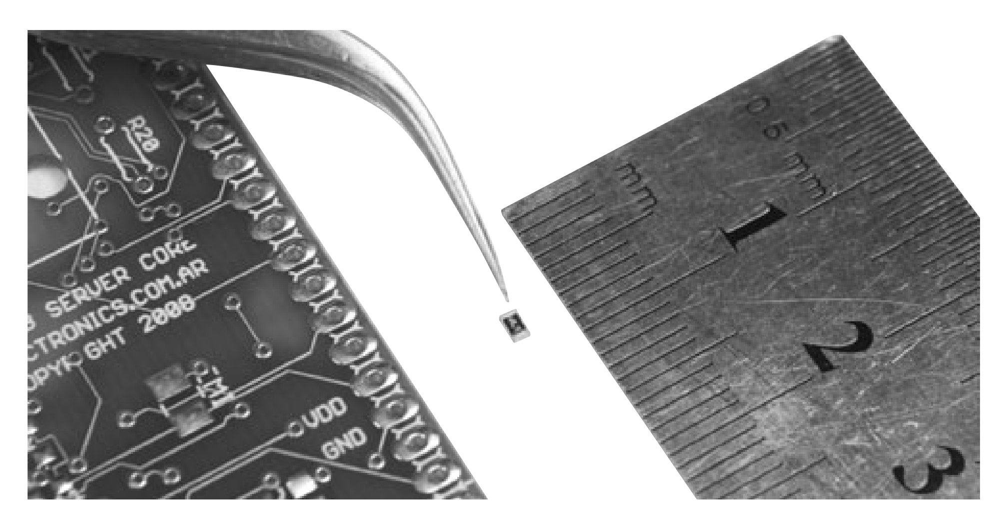
SUBFAMILIA	TIEMPO DE PROPAGACIÓN (NSEG)	VELOCIDAD DE CONMUTACIÓN (MHZ)	CONSUMO DE POTENCIA POR COMPUERTA (MW)
TTL estándar (54/74)	10	35	10
TTL de bajo consumo (54L/74L)	33	3	1
TTL de alta velocidad (54H/74H)	6	50	22
TTL Schottky (54S/74S)	3	125	20
TTL Schottky de bajo consumo (54LS/74LS)	10	45	2

TABLA 6. **Alguno de los integrantes más importantes de la familia TTL, con sus características de velocidad de conmutación, tiempos de propagación y consumo de potencia.**

FAMILIA CMOSSUBFAMILIA	TIEMPO DE PROPAGACIÓN (NSEG)	CONSUMO DE POTENCIA POR COMPUERTA @ 1 MHZ (MW)
Serie 4000	35	0,60
CMOS de alta velocidad (HC)	35	0,06
CMOS de alta velocidad compatible con TTL (HCT)	35	0,06
CMOS avanzado (AC)	35	0,75
CMOS avanzado compatible con TTL (ACT)	35	0,75

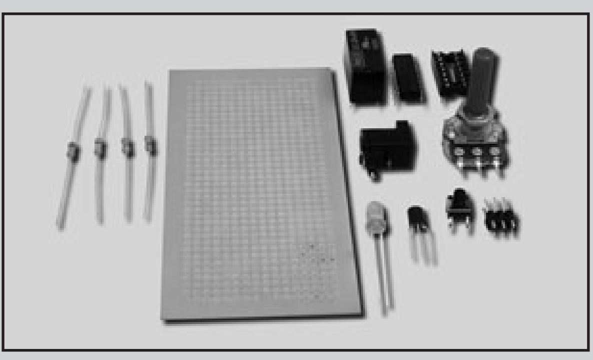
TABLA 7. Algunos de los integrantes más importantes de la familia CMOS, con sus características de consumo de potencia y tiempos de propagación.

Podemos definir una familia lógica como una estructura base a partir de la cual es posible construir diversas arquitecturas lógicas. Dicha estructura base involucra a todos los componentes con los que están constituidas las compuertas lógicas. Las arquitecturas lógicas a las que nos referimos están formadas por elementos discretos, tales como transistores, resistores y diodos, entre otros. Como estamos hablando de electrónica digital, no debemos descuidar el hecho de que las señales pueden tomar dos estados bien definidos: **alto** o **bajo**.



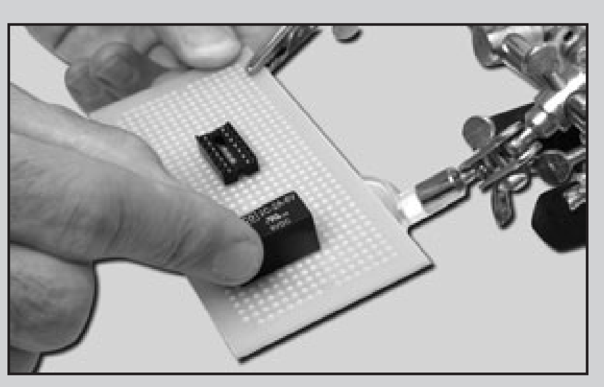
PASO A PASO /1
Construir un temporizador con 4093





Debe disponer de todos los componentes necesarios para el armado del circuito, verificando sus valores con el circuito esquemático. Asimismo, utilice las herramientas adecuadas para facilitar la inserción y posterior soldadura de todos ellos: un soldador con punta cerámica, una pinza de punta fina y un alicate resultarán de gran utilidad.

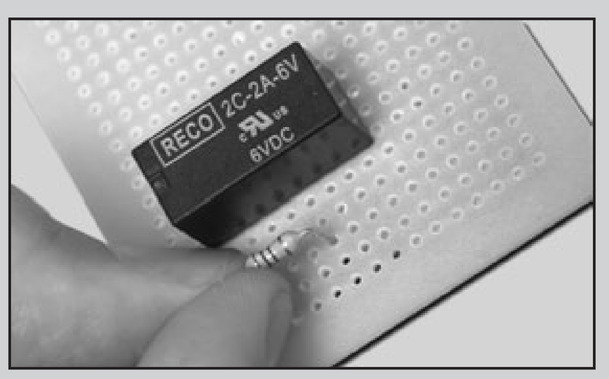




Presente en la placa los componentes más grandes para lograr su mejor ubicación, verificando el circuito (deben estar separados unos de otros). Una vez ubicados, debe soldar los pines por la parte de atrás de la placa, cuidando siempre que el componente quede al ras y nunca elevado de la placa. Aún no coloque el circuito integrado.

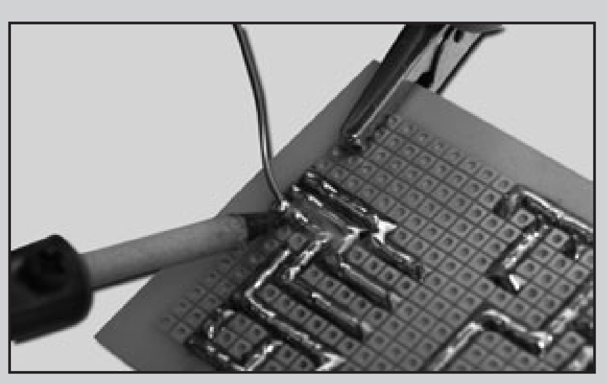
PASO A PASO /1 (cont.)





Una vez ubicados los componentes más grandes, continue con la disposición de los más pequeños. Solde los resistores, el transistor, el capacitor, el LED, el pulsador y el conector de pines. Utilice la pinza de punta para doblar y acomodar los alambres de los componentes de manera que quede prolijo, y el alicate, para cortar los alambres.

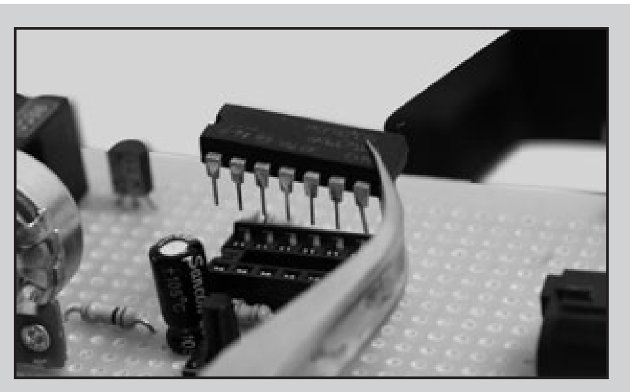




Debe observar el circuito y realizar las uniones entre los componentes con cables o con estaño, como en este caso. Es preciso cuidar que las pistas de estaño queden bien unidas entre sí y que no haya soldaduras "frías" o mal hechas, porque esto puede hacer que no conduzcan. También conviene evitar el exceso de estaño y la unión entre pistas diferentes.

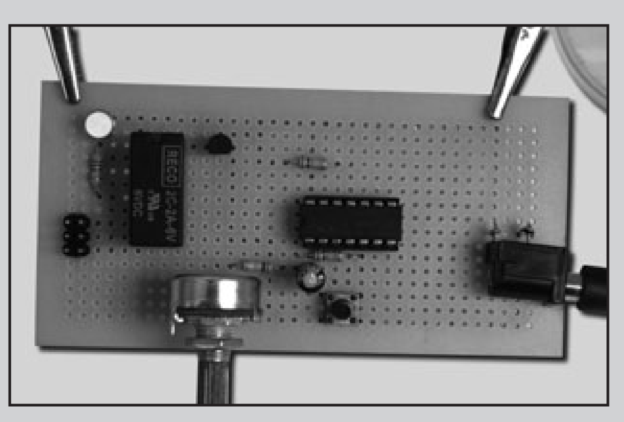
PASO A PASO /1 (cont.)





Utilice una pinza de Bruselas para colocar el circuito integrado en el zócalo correspondiente. En este punto, debe prestar atención a la ubicación del pin n° 1 del zócalo y del circuito integrado. Verifique que los componentes estén bien soldados y utilice un multímetro para comprobar que no haya continuidad entre las pistas de alimentación (VCC y GND).





Conecte el circuito a la fuente de alimentación. Según el relé que utilice, puede ser 5 V o 12 V. Hay que utilizar un relé acorde; es recomendable uno de 12 V, de los empleados en paneles de alarma. Lleve el potenciómetro P1 a su posición media y presione el pulsador para dar inicio al ciclo de temporización.

Por lo tanto, los circuitos que componen las compuertas lógicas deben tener dos regiones de operación bien establecidas y diferenciadas entre sí.

Las familias lógicas pueden clasificarse de acuerdo con el tipo de elemento semiconductor en el cual se basan. Tenemos, entonces, **familias bipolares,** que utilizan transistores bipolares, diodos y resistores; y **familias MOS,** que emplean transistores de efecto de campo MOS. Entre otras características, estas familias difieren en los rangos de tensión para los cuales se definen los niveles lógicos de cada una. En la **Figura 23** se observa esta diferencia.

Las familias bipolares se basan en transistores de unión o bipolares. La más representativa y de mayor uso es la **TTL,** dentro de la cual hay diferentes subfamilias que describiremos a continuación:

TTL (Transistor-Transistor Logic)

Introducida en los años 60, esta fue la familia más utilizada en dispositivos **SSI** y **MSI** (pequeña y mediana escala de integración, respectivamente), y en la actualidad está siendo desplazada por la **CMOS**. Existen diversas subfamilias **TTL**, cada una de las cuales tiene una característica particular: **S** (Schottky), **LS** (Schottky de bajo consumo), **AS** (Schottky mejorado), **ALS** (Schottky mejorado de bajo consumo), **F** (alta velocidad) y **L** (bajo consumo), entre las más relevantes, ver **Tabla 6**.

Familias MOS

Utilizan transistores de efecto de campo como elementos de conmutación. La más representativa de la familia es la CMOS, dentro de la cual existen diferentes subfamilias descriptas a continuación:



CMOS (Complementary Metal Oxide Semiconductor): actualmente está desplazando a la familia TTL en dispositivos **SSI** y **MSI** (pequeña y mediana escala de integración, respectivamente), debido a sus superiores características de velocidad, potencia disipada, márgenes de ruido y fanout. Existen diversas subfamilias CMOS, cada una de ellas con una característica especial: **HC** (alta velocidad), **AHC** (alta velocidad avanzado), **AC** (avanzado), **HCT** (alta velocidad compatible con TTL) y **FACT** (alta velocidad avanzado compatible con TTL), entre las más relevantes, ver **Tabla 7**.

TEMPORIZADOR CON 4093

Luego de haber leído abundante teoría, en el **Paso a paso 1** describimos los pasos para construir un temporizador. El **CD4093B** pertenece a la familia **CMOS** y contiene cuatro compuertas **NAND** de dos entradas cada una, del tipo **Schmitt-Trigger**. Esto permite fijar dos umbrales bien definidos para el cambio de estado. Según vemos en el circuito, el circuito en reposo mantiene a IC1B en 1 mediante la realimentación, con el capacitor descargado. El relé mantiene NA abierto y NC cerrado con COM. Al presionar el pulsador, IC1A cambia a **1**, y esto pone a IC1B en **0**, que mediante la realimentación mantiene a IC1A en **1**, hasta que se cargue el capacitor. Cuando esto ocurre, IC1B cambia otra vez a **1**, poniendo a IC1A en **0** y el capacitor se descarga.

Mientras el ciclo dure, el transistor **T1** estará polarizado y conduciendo, el **LED** se encenderá y el relay cerrará NA con COM.

Multiple **choice**

1 ¿Qué base tiene el sistema de representación posicional decimal?

a- 2.
b- 5.
c- 10.
d- 16.

2 ¿Cuál de los siguientes no es un sistema de representación posicional?

a- Decimal.
b- Binario.
c- Hexadecimal.
d- Ninguno de los anteriores.

3 ¿Cuál de los siguientes sistemas se utiliza de base en los sistemas digitales?

a- Decimal.
b- Binario.
c- Hexadecimal.
d- Ninguno de los anteriores.

4 ¿Cuántos símbolos tiene el sistema hexadecimal para representar números?

a- 2.
b- 5.
c- 10.
d- 16.

5 ¿Cuál de las siguientes unidades de almacenamiento es mayor?

a- Terabyte.
b- Gigabyte.
c- Megabyte.
d- Kilobyte.

6 ¿Cuál de las siguientes no forma parte de las tres compuertas lógicas fundamentales de un circuito combinacional?

a- If.
b- And.
c- Or.
d- Not.

Respuestas: 1 c, 2 d, 3 b, 4 d, 5 a, 6 a.

Capítulo 2

Electrónica Digital



Analizaremos los conceptos
y los dispositivos fundamentales
en los que se basa la Electrónica Digital.

Aritmética **binaria**

A lo largo de este capítulo, analizaremos en profundidad varios conceptos y dispositivos fundamentales en los que se basa la electrónica digital. Esta es una rama muy importante de la electrónica, la cual se basa en el sistema de numeración binario para representar dos estados o niveles lógicos posibles.

En la electrónica analógica, las señales pueden tomar infinitos valores a lo largo del tiempo. En cambio, en la digital, las señales toman sólo dos estados, representados con el **1** y el **0**.

En el transcurso de las siguientes páginas, aplicaremos los conceptos de la aritmética binaria para realizar operaciones con números binarios; posteriormente, emplearemos estos conceptos para realizar un circuito sumador binario.

Analizaremos luego las diferencias entre **lógica combinacional** y **secuencial**. Veremos, además, **biestables** (**flip-flops**), **contadores**, **registros** y **multiplexores**, y realizaremos en forma práctica un **secuenciador** y un **simulador** de vela/hogar a leña.

En el **Capítulo 1** hicimos una primera aproximación a las operaciones con números binarios. Suma, resta, multiplicación y división son operaciones aritméticas que solemos realizar con números decimales. A continuación, veremos cómo hacerlas empleando los números binarios.

Al operar con números binarios, debemos tener en cuenta que si el resultado no está dentro del rango que va desde **0** hasta **2n-1**, siendo **n** la cantidad de bits, no cabrá dentro de la precisión que estamos usando.

SUMA DE NÚMEROS BINARIOS

La suma se realiza bit a bit. Por ejemplo, para sumar dos números binarios, se procede sumando **dígito** a **dígito** y transportando el **acarreo** (si existe) al dígito de mayor peso, es decir, el inmediato izquierdo. Veamos un ejemplo para el caso de z en la **Figura 1**.

Podemos verificar el resultado de la operación anterior realizándola en decimal: vemos que **10 + 4 = 14. Si hacemos la conversión de un sistema a otro, comprobaremos que ambos resultados coinciden.** Veamos otro ejemplo en la **Figura 2**.

```
  1010
+ 0100
  ----
  1110
```

FIGURA 1. **Este resultado no posee** acarreo (carry) final**, con lo cual cabe dentro de la precisión que estamos usando.**

```
  1110
+ 1000
 -----
 10110
```

FIGURA 2. **Este resultado posee acarreo final, con lo cual no cabe dentro de la precisión que estamos usando.**

RESTA DE NÚMEROS BINARIOS

La **resta** también se realiza bit a bit. Una forma sencilla de hacerla es sumarle al minuendo el complemento a 2 del sustraendo. Veamos un ejemplo, en la **Figura 3**, para el caso de **4 bits**.

```
  1001
- 0100
------
```

Lo anterior es equivalente a

```
  1001
+ 1100
------
 10101
```

FIGURA 3. La presencia de acarreo final en este caso está indicando que no existió préstamo (borrow) en el resultado, con lo cual cabe dentro de la precisión que estamos usando.

MULTIPLICACIÓN DE NÚMEROS BINARIOS

Para multiplicar dos números binarios, se procede de igual forma que en el sistema decimal: se multiplica el **multiplicando** por cada dígito del **multiplicador**, comenzando por el de menor peso, y finalmente se suman los resultados con su peso correspondiente. Una manera sencilla de hacerlo es tomando el **bit menos significativo del multiplicador**. Si es **1**, tomamos el multiplicando como un primer resultado, y si es **0**, tomamos **0** como primer resultado. Luego, desplazamos el multiplicando un dígito hacia la izquierda (recordemos que desplazar una cifra binaria un lugar hacia la izquierda equivale a multiplicar por 2). A continuación, tomamos el segundo bit de la derecha y hacemos lo mismo; sumamos este nuevo resultado al anterior. Repetimos así esta operación hasta acabar con todos los dígitos del multiplicador. Estas operaciones se simplifican con el uso de registros de desplazamiento, tema que explicaremos más adelante. Veamos un ejemplo en la **Figura 4**.

```
     1101
x     101
---------
     1101
+   0000
+ 1101
---------
  1000001
```

FIGURA 4. Multiplicación con números binarios.



CALCULADORA DE WINDOWS

Para convertir un sistema de numeración en otro, primero seleccionamos el sistema de origen (**Hexa**, **Dec**, **Oct** o **Bin**), escribimos el número por convertir y, luego, hacemos clic sobre el sistema de numeración al que deseamos llevarlo.

DIVISIÓN DE NÚMEROS BINARIOS

Para la división, podemos aplicar un algoritmo sencillo. La operación comienza preparando una resta, en la que ubicamos como minuendo al dividendo, y como sustraendo, el divisor, pero colocado a la izquierda, de forma tal que el peso del bit más significativo del divisor coincida con el del bit más significativo del dividendo. Si vemos que en esa posición el minuendo es menor al sustraendo, no se realiza la resta, y colocamos un **0** como dígito más significativo del cociente. Si no, hacemos la resta y ponemos un **1** como dígito más significativo del cociente. En el siguiente paso, el resultado de la resta anterior será el nuevo minuendo (si no hubo resta, se mantendrá). Luego, el nuevo sustraendo será igual al anterior, pero desplazado un dígito hacia la derecha. Procedemos de la misma manera, anotando el nuevo resultado (si corresponde) y agregando un nuevo dígito al cociente (un dígito menos significativo que el anterior). Esta operación se lleva a cabo hasta que el nuevo sustraendo coincida con el dividendo. Veamos un ejemplo en las **Figuras 5**, **6**, **7** y **8.**

Dividendo: 10110
Divisor: 10

```
  10110
- 10000
  -----
  00110
```

FIGURA 5. Se realizó una resta. El dígito más significativo del cociente es un 1.

```
  00110
- 01000
  -----
  00110
```

FIGURA 6. No se realizó la resta. El dígito inmediato derecho del cociente es un 0.

```
  00110
- 00100
  -----
  00010
```

FIGURA 7. Se realizó una resta. El dígito inmediato derecho del cociente es un 1.

```
  00010
- 00010
  -----
      0
```

FIGURA 8. Se realizó una resta. El dígito menos significativo del cociente es un 1 y el resto, 0.

Finalmente, en el ejemplo obtuvimos:
Cociente: **1011**
Resto: **0**



PRESENTACIÓN EN LA CALCULADORA

Dentro de las opciones de la calculadora, podemos seleccionar el tamaño de presentación que deseamos utilizar (**Qword**, **Dword**, **Word** o **Byte**). La desventaja es que esta calculadora no nos informa acerca de la existencia de acarreo.

CIRCUITO SUMADOR BINARIO DE 1 BIT CON ACARREO

Para realizar una suma binaria de 1 bit, debemos tener en cuenta el resultado para la suma de **1 + 1** (caso especial), en la cual se obtiene 10 (resultado 0 y acarreo 1). Este circuito constará de dos entradas que llamaremos **A** y **B**; y de dos salidas, las cuales serán **S** para el resultado (suma) y **C** para el acarreo (carry). Sobre esta base, podemos confeccionar la **Tabla 1**.

Vemos que la compuerta necesaria para obtener **S** es una **XOR**, y la requerida para obtener **C** es una **AND**. En ambas, ingresan las entradas **A** y **B**. Este sencillo circuito se denomina medio-sumador, y solo puede sumar cifras de un dígito. Si queremos sumar cifras de n bits, necesitamos un sumador completo, como se observa en la **Figura 9**. ¿Cómo funciona?

LÓGICA COMBINACIONAL Y SECUENCIAL

Los circuitos lógicos pueden tener una o más entradas y una o más salidas, las cuales presentarán dos niveles de tensión posibles: el correspondiente para el **0 lógico** y el correspondiente para el **1 lógico**. Además, todo circuito lógico puede clasificarse en dos tipos: **combinacional** o **secuencial**.

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

TABLA 1. Circuito sumador binario de 1 bit con acarreo.

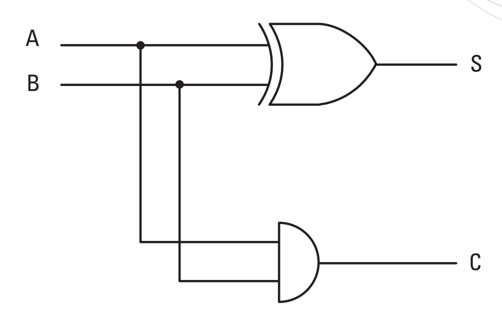


FIGURA 9. A la izquierda vemos un circuito medio-sumador binario de 1 bit con acarreo. A la derecha, un sumador completo de 1 bit con acarreo.

CIRCUITOS LÓGICOS COMBINACIONALES

Ya hemos aprendido que en los circuitos lógicos combinacionales el estado de las salidas es función exclusiva del estado de las entradas en un instante determinado. También estudiamos las compuertas: **OR**, **AND**, **NOR**, **NAND**, **XOR**, **XNOR**, **inversor** y **buffer**. Pertenecen también a la categoría de circuitos lógicos combinacionales los dispositivos integrados, tales como: **comparadores de magnitud**, **sumadores**, **codificadores**, **decodificadores**, **multiplexores** y **demultiplexores**, entre otros.

CIRCUITOS LÓGICOS SECUENCIALES

En los circuitos lógicos **secuenciales**, el estado de las salidas es función no solo del estado de las entradas en ese mismo instante, sino también del estado previo de ellas, con lo cual almacenan (memorizan) los estados anteriores. Existen dos tipos de circuitos lógicos secuenciales. Por un lado, los **asincrónicos**,

que poseen salidas realimentadas hacia las entradas. Esto permite que los estados de las salidas sean función no solo de los estados de las entradas en ese instante, sino también de los estados anteriores de estas. Por otro lado, están los **sincrónicos**, en los cuales los cambios de estado se producen en sincronismo mediante una señal de reloj (**clock**).

CIRCUITOS CON REALIMENTACIÓN

Analizaremos los diferentes circuitos con realimentación; en particular, veremos los **biestables** o **flip-flops**. Hay flip-flops en los que los cambios de estado se producen en forma sincrónica con una señal de reloj. Existen los disparados por nivel y los disparados por flanco. Los segundos pueden dividirse en dos tipos: por flanco positivo y por flanco negativo.

En los flip-flops sincrónicos, los cambios de estado se producen en forma sincrónica con una señal de reloj

S	R	Q
0	0	no varía
0	1	0
1	0	1
1	1	no deseado

TABLA 2. **Características del flip-flop.**

FLIP-FLOP RS (O SR)

Este circuito secuencial posee dos entradas llamadas **S** (set) y **R** (reset); y dos salidas llamadas **Q** y **\Q**, siendo **\Q** el complemento o negación lógica de **Q**. Al poner un **1** a la entrada **S**, el estado de la salida **Q** también se pondrá en **1**.

En cambio, al poner un **1** en la entrada **R**, el estado de esta salida se pondrá en **0**. Cuando ambas entradas tienen nivel lógico **0**, el estado de las salidas podrá ser **0** o **1,** dependiendo del estado anterior de las entradas. En caso de que el estado de **S** y **R** sea **1,** producirá una indeterminación en la salida, con lo cual queda como estado **no deseado**. Veamos la tabla característica de este flip-flop (**Tabla 2**).

Si modificamos un poco este circuito agregándole dos compuertas **AND**, podemos añadirle una entrada de habilitación **LE** (Latch Enable). De este modo, cuando esta entrada **LE** esté en estado lógico **1**, el flip-flop funcionará normalmente. Cuando esté en estado **0**, el flip-flop mantendrá el último valor de su salida, sin importar el de las entradas **S** y **R** aplicados posteriormente. La tabla característica de este flip-flop con el agregado de la entrada de habilitación es la **Tabla 3**.

S	R	R	Q
0	X	X	no varía
1	0	0	no varía
1	0	1	0
1	1	0	1
1	1	1	no deseado

TABLA 3. **Tabla característica del flip-flop con el agregado de la entrada de habilitación.**

LATCH (FLIP-FLOP D ASINCRÓNICO)

Este circuito secuencial permite que la salida **Q** siga los cambios de la entrada **D**, siempre y cuando esté habilitada la entrada **LE**. La tabla característica de este flip-flop es la **Tabla 4**:

LE	D	Q
0	X	no varía
1	1	0
1	1	1

TABLA 4. Tabla de Flip-flop D asincrónico**.**

FLIP-FLOP RS SINCRÓNICO

El funcionamiento de este **flip-flop RS** es análogo al del asincrónico, con la diferencia de que los cambios de estado se producen en forma sincrónica con una señal de reloj, que es ingresada por la entrada **CLK** del flip-flop. La tabla característica de este flip-flop es la **Tabla 5**:

S	R	Q*
0	0	Q
0	1	0
1	0	1
1	1	X

TABLA 5. Características del Flip-flop RS sincrónico.

Se denomina **Q*** al valor que tomará la salida del flip-flop en el siguiente flanco correspondiente de la señal de reloj. Aquí también vemos que si el estado de las entradas **S** y **R** es **1,** se produce una indeterminación en la salida, con lo cual, al igual que en el **flip-flop RS asincrónico**, queda como estado no deseado.

Además de las entradas **S**, **R** y **CLK**, existen dos más llamadas **PRE** (preset) y **CLR** (clear). La primera fuerza la salida **Q** al estado **1**, mientras que la segunda la fuerza al estado **0**.

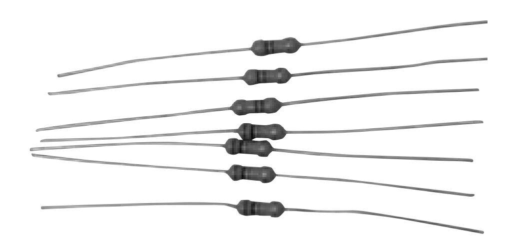
FLIP-FLOP JK

Se comporta de manera similar al **RS**, con la excepción de que en este último se permite que ambas entradas, **J** y **K,** estén en estado **1** al mismo tiempo. La tabla característica de este flip-flop es la **Tabla 6**.

J	K	Q*
0	0	Q
0	1	0
1	0	1
1	1	\Q (Q negado)

TABLA 6. Características del Flip-flop JK**.**

Sobre la base de esta tabla, podemos ver que el **flip-flop JK** cubre todas las características del **RS**. Cuando las entradas **J** y **K** están en estado **1** al mismo tiempo, el estado futuro **Q*** será el opuesto al anterior, con lo cual será **Q (Q negado)**. Las entradas de **CLK**, **PRE** y **CLR** son las mismas que para el **flip-flop RS**.



FLIP-FLOP T (TOGGLE)

Fácilmente, podemos obtener un **flip-flop T** con solo unir las entradas **J** y **K** de un **flip-flop JK**, y llamando **T** a esta única entrada. La tabla característica de este flip-flop es la **Tabla 7**.

T	Q*
0	Q
1	\Q (Q negado)

TABLA 7. Características del flip-flop T.

Sobre la base de la tabla anterior, podemos describir el funcionamiento del **flip-flop T** de la siguiente manera: cuando la entrada **T** tiene nivel lógico **0**, el estado de la salida **Q,** en el próximo flanco correspondiente de la señal de reloj (el estado de **Q***) no se modifica. En cambio, si la entrada **T** tiene nivel lógico **1**, el estado de la salida **Q** en el próximo flanco correspondiente de la señal de reloj se invierte.

Una de las aplicaciones que puede tener el **flip-flop T** es como divisor de frecuencia, ya que si aplicamos un **1**a la entrada **T**, la salida **Q** irá alternando su estado en cada flanco correspondiente de la señal de reloj. Con esto, estaríamos dividiendo la frecuencia del reloj a la mitad.

El flip-flop T se utiliza como divisor de frecuencia. El D, como demora de un clock

FLIP-FLOP D (DELAY)

Este flip-flop permite que la salida **Q** siga los cambios de la entrada **D**, siempre en cada flanco correspondiente de la señal de reloj. La tabla característica de este flip-flop es la **Tabla 8**.

Se logra a partir de un flip-flop **JK** uniendo sus entradas con un inversor, de modo que **J = D** y **K = \D (D negado)**.

A su vez, si en un flip-flop **D** se conecta **D = \Q (Q negado)**, se lo transforma en un flip-flop **T**.

D	Q*
0	Q
1	1

TABLA 8. Características del flip-flop D (delay).

Circuitos secuenciales

Ya sabemos qué son los flip-flops y cómo funcionan. Ahora veremos de qué manera se utilizan para constituir la base de los circuitos secuenciales, contadores sincrónicos y asincrónicos.

Los contadores son circuitos lógicos secuenciales que llevan la cuenta de una serie de pulsos de entrada. Se forman a partir de **flip-flops** conectados entre sí y combinados muchas veces con compuertas lógicas. En los sistemas digitales, se utilizan variaciones de estos circuitos en forma de circuitos integrados.

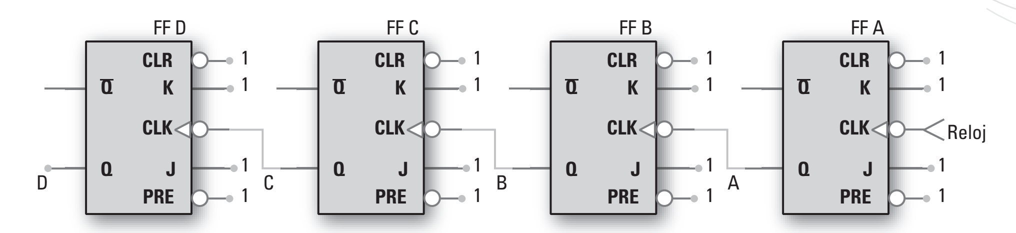


FIGURA 10. Contador asíncrono de 4 bits. Se lo llama de este modo porque los flip-flops no modifican su estado en sincronía exacta con los pulsos de reloj.

Podemos clasificar los contadores como **binarios** o no **binarios** y **síncronos/sincrónicos** o **asíncronos/asincrónicos**. Aquí nos ocuparemos de los binarios, llamados así porque el conteo se realiza en código binario; además, veremos los del tipo síncrono y asíncrono. Analizaremos ahora la estructura de ambos.

CONTADOR BINARIO ASÍNCRONO

Para entender el funcionamiento de este tipo de contador veamos el circuito **Contador asíncrono de 4 bits**. Se trata de un contador binario de 4 bits formado por cuatro flip-flops **JK** conectados en serie. Los pulsos de reloj se aplican únicamente a la entrada de reloj (**CLK**) del primer flip-flop. Como las entradas **JK**

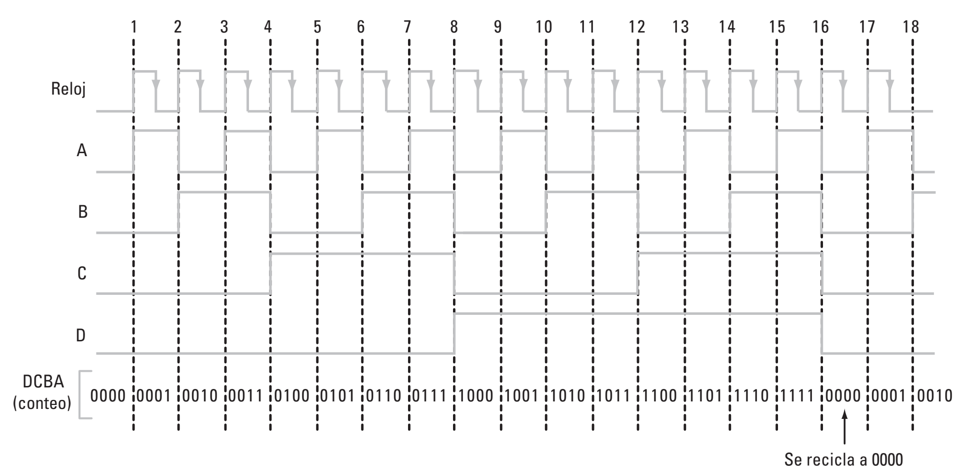


FIGURA 11. Las formas de onda de salida de cada flip-flop y de la señal de reloj. Vemos una secuencia de conteo de 0000 a 1111. En la decimosexta transición negativa de reloj se inicia un nuevo ciclo de conteo.

Los contadores son circuitos lógicos secuenciales que llevan la cuenta de una serie de pulsos de entrada

del primer flip-flop (**flip flop A**) tienen un valor igual a **1** (nivel alto), cada vez que los pulsos de reloj hagan una transición negativa (de 1 a 0), el flip-flop pasará a su estado opuesto. Este modo de funcionamiento corresponde al de un **flip-flop T** (toggle). La salida normal del **flip-flop A** es la entrada de CLK para el **flip-flop B**. Por lo tanto, el flip-flop B cambiará su estado cada vez que la salida de A pase de **1** a **0**. De esta misma manera, el **flip-flop C** cambiará de estado cuando **B** pase de **1** a **0**, y así sucesivamente. Si ahora observamos las señales del circuito, notaremos que las salidas D, C, B y A del flip-flop representan un número binario de 4 bits (**Figuras 10** y **11**).

En el contador asíncrono, cada salida del flip-flop excita la entrada CLK del siguiente flip-flop. Por lo tanto, es un contador **asíncrono,** porque los flip-flops no modifican su estado en sincronía exacta con los pulsos de reloj. Concluimos, entonces, que hay un retardo entre las respuestas de los flip-flops, generalmente, de entre 5 y 20 **ns** (nanosegundos) por cada uno.

MÓDULO DE UN CONTADOR

El contador que acabamos de ver tiene 16 estados diferentes (0000 a 1111); su módulo es 16 (MOD-16). Si queremos aumentarlo, simplemente agregamos flip-flops. Entonces, **MOD=2N**, donde **N** es el número de flip-flops conectados. Un contador actúa como un divisor de frecuencia, ya que la forma de onda de salida del último flip-flop tendrá una frecuencia que será igual a la frecuencia de entrada de reloj dividida por el módulo del contador.

El contador visto anteriormente posee un valor de módulo sujeto a la expresión **2N**. Por ejemplo, si deseamos construir un contador con módulo igual a **10**, que no es una potencia de **2**, debemos buscar la manera de omitir estados en la secuencia de conteo. Para hacerlo, se utilizan compuertas lógicas.

Tomamos un contador de 4 bits, es decir, de cuatro flip-flops. Este sería, entonces, de módulo **16** si omitimos la compuerta **NAND**. Si ahora tomamos en cuenta su función, vemos que cuando sus entradas **B** y **D** sean iguales a **1**, su salida será igual a **0** y, al estar conectada a la entrada **CLR** de los flip-flops, estos se borrarán y el contador pasará al estado **0000**. Esto ocurrirá justamente cuando el contador pase del estado **1001** al estado **1011** en la transición negativa del décimo pulso de entrada. El contador pasa por diez estados diferentes antes de reciclar su cuenta; se trata de un contador de módulo **10** (**Figuras 12** y **13**).

CONTADOR BINARIO SÍNCRONO

El contador asíncrono tiene el inconveniente de poseer retardos de propagación en su funcionamiento, por lo que no recomendamos su uso a frecuencias altas. Podemos superar este inconveniente utilizando contadores **síncronos**. Veamos el caso de un **contador síncrono de módulo 16** (**Figura 14).** Para un conteo adecuado, solo

los flip-flops que deben cambiar de estado necesitan tener sus entradas **J** y **K** iguales a **1** cuando ocurra una transición negativa en el pulso de reloj. El flip-flop **A** tiene sus entradas permanentemente en **1,** por lo que cambia su estado con cada transición negativa de la entrada de reloj.

El **flip-flop B** debe cambiar estados en cada transición negativa de reloj que ocurra mientras **A = 1**. Esto se logra conectando la salida **A** a las entradas **J** y **K** del flip-flop B. **El flip-flop C** debe cambiar estados en cada transición negativa de reloj que ocurra mientras **A = B = 1**. De la misma forma, el **flip-flop**

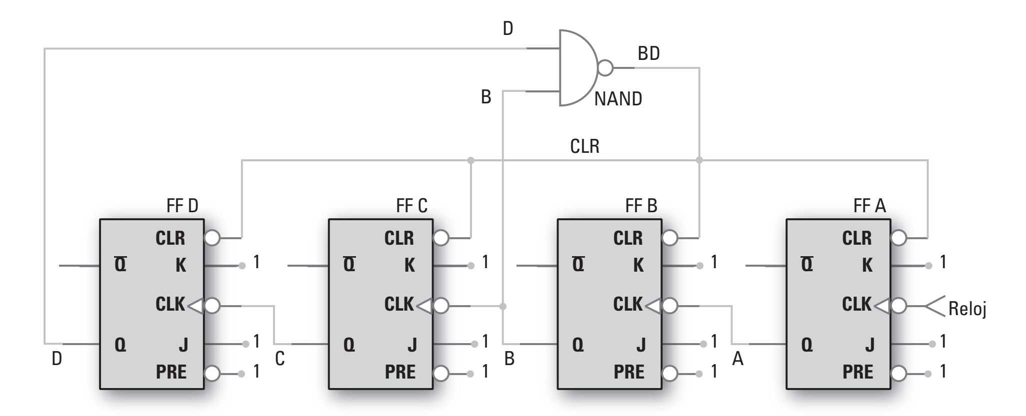


FIGURA 12. Un contador asíncrono de módulo 10. La acción de la compuerta NAND permite omitir estados en la secuencia. Podríamos implementarlo utilizando un integrado 74LS293 TTL.

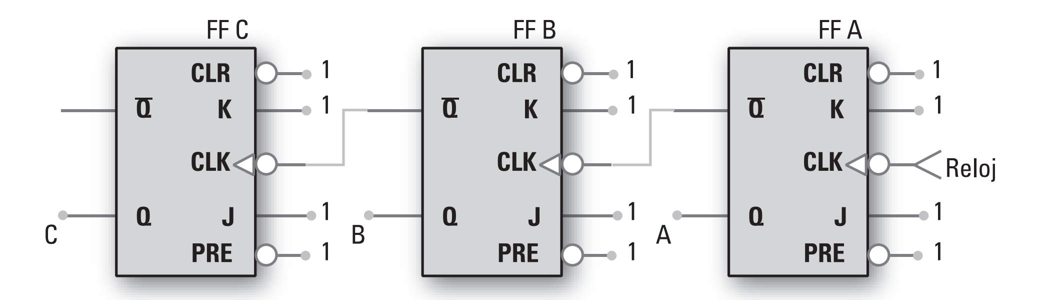


FIGURA 13. Hasta ahora analizamos contadores de 0 hacia arriba (ascendentes). Para un contador descendente, los flip-flops se conectan a través de su salida negada. Vemos aquí un contador de módulo 8.

D tiene que cambiar estados en cada transición negativa de reloj que ocurra mientras **A = B = C = 1**. Para realizar todas estas operaciones se utilizan compuertas **AND**.

La mayoría de los circuitos integrados comerciales que funcionan como contadores síncronos se diseñan de modo tal que se pueda preestablecer el valor inicial del conteo de forma **asíncrona** (independiente de la señal de reloj) o de forma **síncrona** (en la transición activa de la señal de reloj). El contador posee entradas exclusivas para el preestablecimiento que se realiza de manera paralela. Además, por lo general tienen una entrada de control que establece si la cuenta será ascendente o descendente, y salidas de acarreo y préstamo para conectar más de un contador y aumentar el módulo (**Figura 15**).

En un contador síncrono, los flip-flops se disparan al mismo tiempo, mediante los pulsos de entrada de reloj

CONTADOR EN ANILLO Y CONTADOR JOHNSON

Analizaremos en este apartado los contadores que emplean realimentación en su funcionamiento, ya que la salida del último flip-flop se conecta a la entrada del primero.

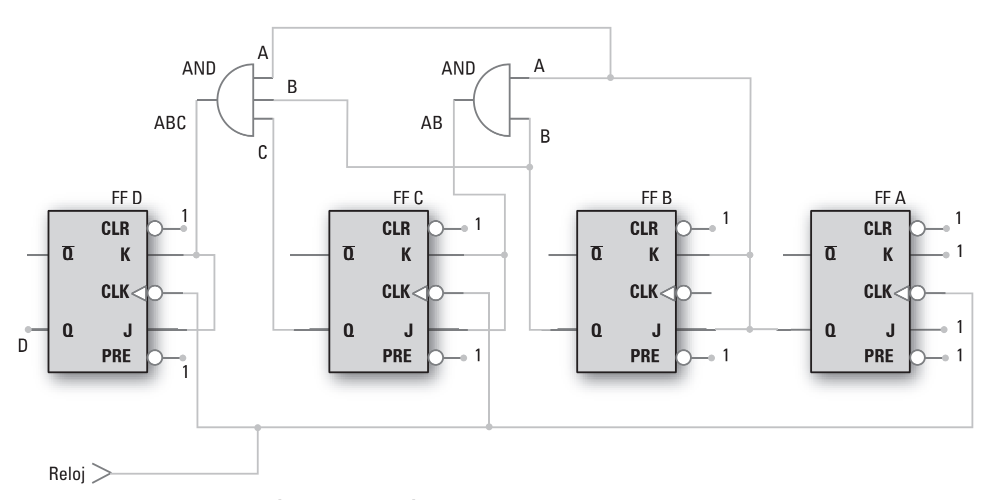


FIGURA 14. Un contador síncrono de módulo 16. Los cambios de estado de los flip-flops están sincronizados con la entrada de reloj. Un circuito integrado comercial de este tipo es el 74HC163 CMOS.

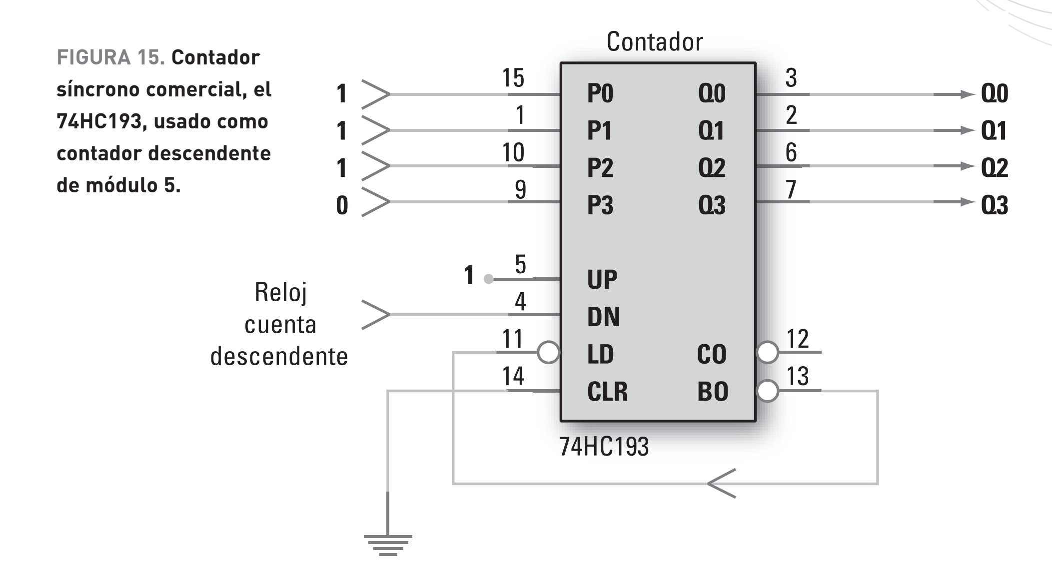


FIGURA 15. Contador síncrono comercial, el 74HC193, usado como contador descendente de módulo 5.

CONTADOR EN ANILLO

Su estructura se basa en flip-flops de tipo **D**, como se observa en la imagen correspondiente. La salida normal del último flip-flop se conecta directamente a la entrada **D** del primero (**Figura 16**). De esta manera, la información se desplaza de izquierda a derecha, y de regreso, de **Q0** a **Q3**. Un valor lógico **1** se hace circular por todo el arreglo de flip-flops cuando se aplican pulsos de reloj. Por esta razón se lo llama **contador en anillo**.

Tomemos como ejemplo un contador en anillo de 4 bits. Si suponemos un estado inicial de **Q3 = 1**, **Q2 = Q1 = Q0 = 0**, vemos en las formas de onda que, después del primer pulso de reloj, el **1** se desplaza de **Q3** a **Q2**, y el estado resultante es **0100**. Con cada pulso, el **1** se desplaza, y en el cuarto pulso, se vuelve al estado inicial. Es un contador de módulo 4 porque atraviesa por cuatro estados diferentes antes de repetir la cuenta (**Figura 17**). El módulo del contador de este tipo es igual a la cantidad de flip-flops que utiliza; por lo tanto, es posible construir contadores en anillo para cualquier valor de módulo. Las formas de onda de los flip-flops están desplazadas un período de reloj una con respecto a la otra, y en cada una de ellas el estado lógico **1** tiene una duración de un período de reloj.

El contador en anillo necesita iniciar su secuencia de conteo con solo un flip-flop en estado lógico **1** y todos los demás, en **0**. Para esto, se aplica un pulso momentáneo a la entrada asíncrona **PRESET** de uno de los flip-flops, y a la entrada **CLR** de todos los otros flip-flops. También pueden utilizarse circuitos externos para asegurar este inicio.

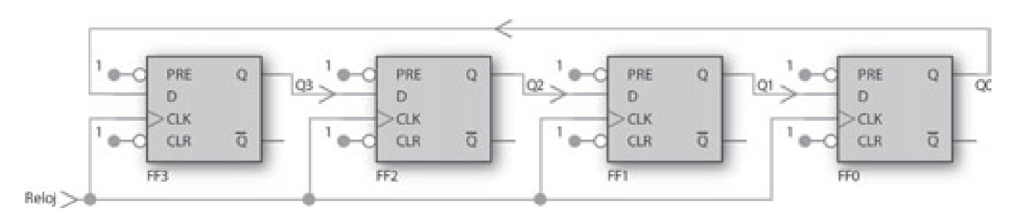


FIGURA 16. Flip-flops del tipo D. Un contador en anillo de 4 bits formado por flip-flops del tipo D. La señal de reloj se aplica simultáneamente a todos los flip-flops, por lo que es un contador síncrono.

1 2 3 4 5 6 7 8

Reloj

Q_3

Q_2

Q_1

Q_0

(b)

Q_3	Q_2	Q_3	Q_3	Pulso de RELOJ
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7
.	.	.	.	.
.	.	.	.	.

(c)

1000

0100

0010

0001

(d)

FIGURA 17. Las formas de onda de salida de cada flip-flop y de la señal de reloj, la tabla de secuencia y el diagrama de estados para un contador en anillo. Como el conteo tiene cuatro estados diferentes, su módulo es 4.

CONTADOR JOHNSON

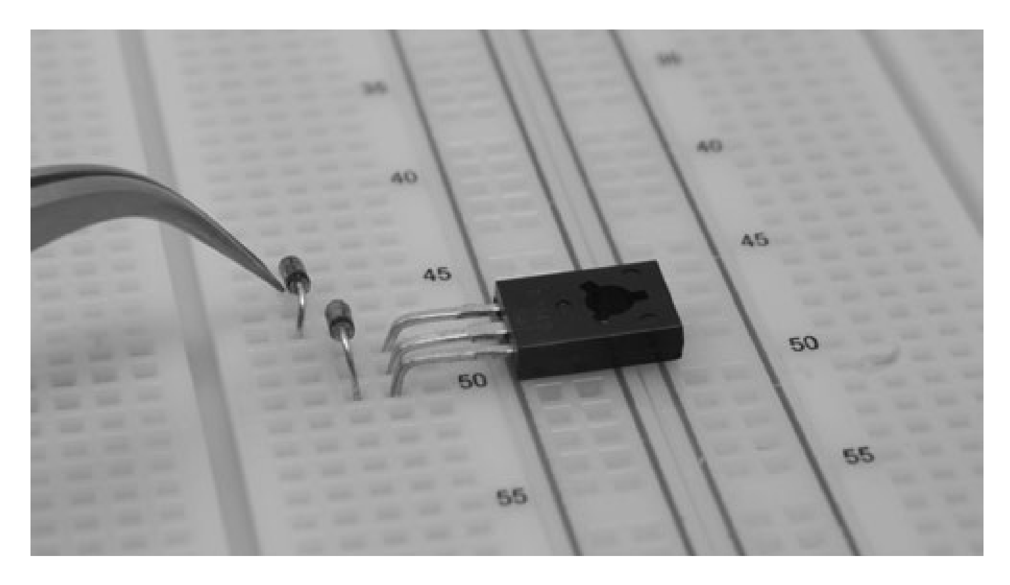
Para realizar este contador, debemos hacer una pequeña modificación al contador en anillo básico. En este caso, la salida invertida del último flip-flop se conecta a la entrada del primero.

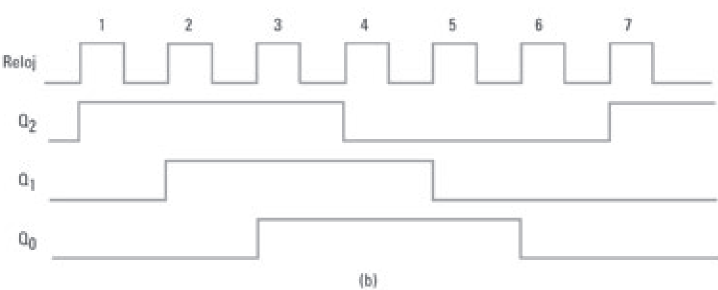
Analicemos un contador **Johnson de 3 bits**. Existe una realimentación debido a que la salida invertida de **Q0** se conecta a la entrada **D** de **Q2**. Esta configuración hace que el nivel inverso del almacenado en **Q0** se transfiera a **Q2** en el pulso de reloj.

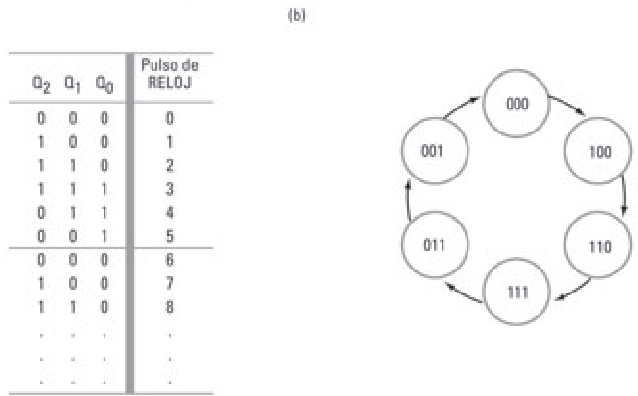
Observando las formas de onda y la tabla de secuencia, deducimos que este contador tiene seis estados diferentes antes de repetir la secuencia. Tarda **3 pulsos** de reloj en propagar el **1** lógico hacia abajo en el arreglo de flip-flops, y otros **3 pulsos** de reloj en regresar todos los flip-flops al estado **nulo**. Concluimos que el valor del módulo de un contador Johnson es igual al doble del número de flip-flops. La forma de onda de cada flip-flop es cuadrada (ciclo de trabajo del 50%) y de una frecuencia igual a la frecuencia de entrada dividida por el módulo del contador. Además, las formas de onda de los flip-flops se desplazan un período de reloj una con respecto a la otra.

Una ventaja del contador Johnson respecto del contador en anillo es que no necesita una rutina de inicialización, siempre y cuando el estado inicial de todos los flip-flops sea nulo.

Los códigos de Johnson son la base de los sistemas de comunicación con corrección de **errores FEC** (*Forward Error Correction*). Su potencia radica en que solo un bit cambia a la vez, lo que permite detectar cambios por efectos del ruido (**Figura 18**).







Q_2	Q_1	Q_0	Pulso de RELOJ
0	0	0	0
1	0	0	1
1	1	0	2
1	1	1	3
0	1	1	4
0	0	1	5
0	0	0	6
1	0	0	7
1	1	0	8
.	.	.	.
.	.	.	.
.	.	.	.

FIGURA 18. **Las formas de onda de salida de cada flip-flop y de la señal de reloj, la tabla de secuencia y el diagrama de estados para un contador Johnson.**

CONTADOR DECIMAL CON DISPLAY DE 7 SEGMENTOS

Ya aprendimos cómo funcionan los contadores. En esta ocasión, construiremos un circuito basado en un contador configurado para una cuenta decimal, la cual se muestra en un display de 7 segmentos (**Paso a paso 1**).

Realizaremos el montaje de un contador decimal basado en el circuito integrado CD4029; éste se configura para operar como un contador decimal con un valor de cuenta inicial igual a 0.

El diseño incluye un decodificador BCD a 7 segmentos (CD4511) que, a su vez, excita un display de 7 segmentos del tipo cátodo común. Incluiremos un conector de dos terminales que nos permitirá aumentar la secuencia de conteo mediante un pulsador externo o emplear directamente un generador de pulsos. Montaremos el diseño en un circuito impreso universal y usaremos una fuente de +5 V.

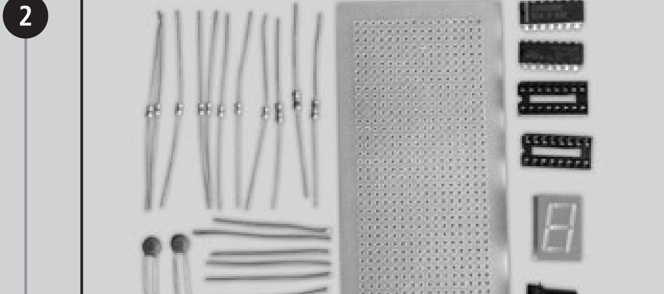
Construiremos un circuito basado en un contador configurado para una cuenta decimal

PASO A PASO /1
Construcción de un contador decimal con display de 7 segmentos

1

Para comenzar, disponga del esquema del diseño realizado mediante Cadsoft Eagle. En él se muestran los pines de alimentación y GND de los circuitos integrados, con el fin simplificar el análisis del diseño. Sin embargo, no debe olvidar conectarlos como correspondan, consultando las hojas de datos de ambos integrados. Como conector de dos terminales (CN1), puede utilizar postes o una bornera bipolar.

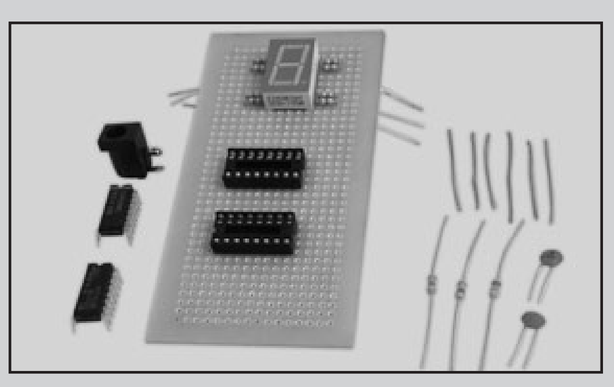
2



Utilizará dos circuitos integrados: el CD4029 y el CD4511, ambos de tecnología CMOS. Colocará los integrados en zócalos para facilitar su montaje y desmontaje. Para conectar la fuente de alimentación, utilizará una clavija hembra de dos polos para circuito impreso. En el diseño, empleará también resistencias y capacitores. El display de 7 segmentos será del tipo cátodo común y alto brillo.

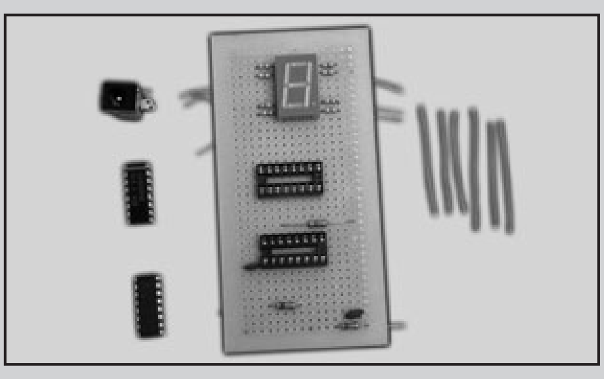
PASO A PASO /1 (cont.)





Coloque el display de 7 segmentos, las 8 resistencias de 470 Ω y los zócalos para los integrados. El display es de tipo cátodo común porque todos los diodos LED internos que iluminan los segmentos tienen sus cátodos unidos entre sí. La salida activa del CD4511 es de +5 V, y con una caída de tensión de +2 V en cada LED, la corriente será de 6 mA.

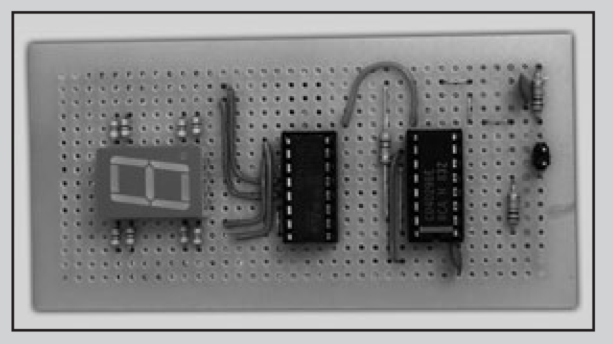




Coloque las resistencias R8, R9 y R13, y los capacitares C1 y C2. El arreglo formado por C1 y R8 permite generar un pulso positivo en la entrada PE del CD4029. Al aplicar este pulso, se ingresa el valor 0 en decimal configurado mediante las entradas P1 a P4 que están en masa. La resistencia R13 de 470 Ohms funciona como protección de la entrada. En este caso, cablee las entradas directamente.

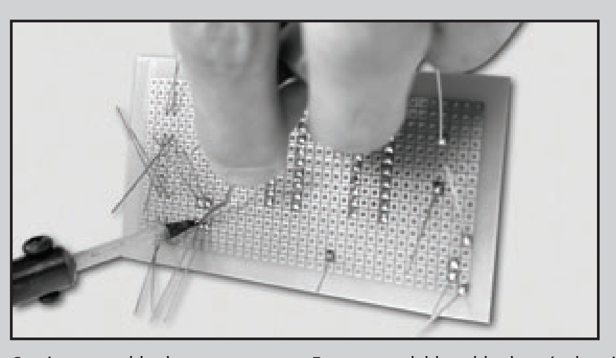
PASO A PASO /1 (cont.)





Coloque los integrados en sus zócalos y utilice puentes de cable para realizar las uniones entre componentes. Agregue los postes. Para conectar un pulsador, se conecta CN1-1 a CN1-2 y se unen a una de las terminales del pulsador. Debe colocar la resistencia R12 como lo indica el esquema. La otra terminal del pulsador se conecta a masa. Para eliminar los rebotes del pulsador, coloque un capacitor de 10 μF en paralelo con este.

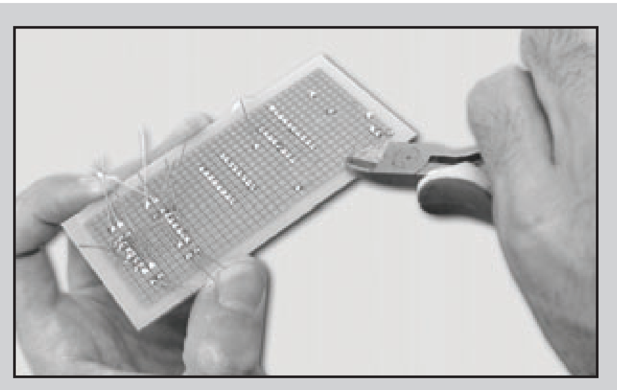




Comience a soldar los componentes. Es recomendable soldar los zócalos sin los circuitos integrados dentro, para evitar dañarlos por sobrecalentamiento. Puede agregar a este diseño una resistencia etiquetada como R10 en el esquema, que limite la corriente de descarga del capacitor C2 y proteja el circuito integrado. Puede incorporar un diodo en antiparalelo con C2 que permita descargar el capacitor y resetear el circuito.

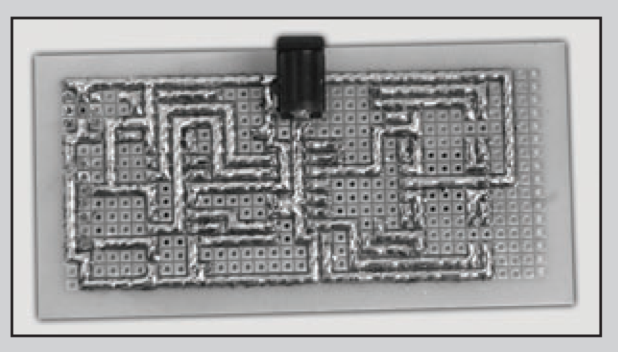
PASO A PASO /1 (cont.)





Use una pinza, corte la parte sobrante de las patas de los componentes para evitar cortocircuitos. Debe hacerlo con cuidado, ya que si fuerza demasiado las patas, podría debilitar la soldadura. Como se observa en la imagen, los circuitos integrados se encuentran ubicados con la misma orientación, para simplificar el conexionado y no cometer errores durante el armado.

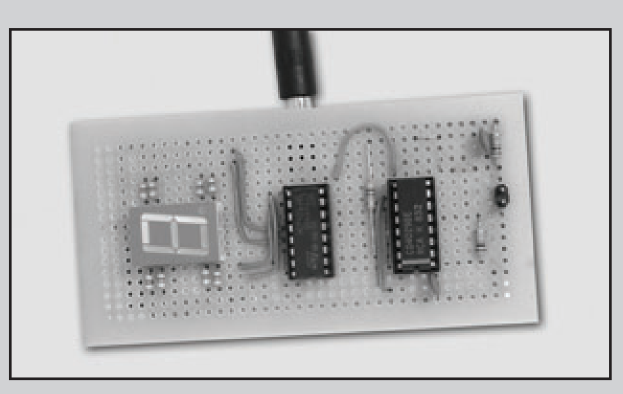




Con el soldador, haga los puentes de estaño necesarios y agregue la clavija hembra de dos polos para circuito impreso. Recuerde que el polo que va por fuera en el conector macho es el que queda más hacia el exterior del circuito impreso. En general, este polo es el negativo de la fuente y la masa del circuito. El que va por dentro del conector macho –y, por lo tanto, más hacia el interior del circuito impreso– es el positivo de la fuente.

PASO A PASO /1 (cont.)





Ya casi está en condiciones de probar el circuito. Lo que debe hacer ahora es valerse del esquema y emplear un multímetro para verificar la continuidad del circuito. Una vez hecho esto, conecte la fuente de +5 V a la clavija, y ya tendrá el circuito en funcionamiento. Este diseño podría armarse sin problemas en un protoboard o en un circuito impreso diseñado en Eagle y montado en una placa de pertinax.

Registros

En este apartado explicaremos qué son y para qué sirven estos dispositivos tan importantes utilizados en la electrónica digital, llamados registros.

Los registros son dispositivos lógicos compuestos, principalmente, por **biestables** o **flip-flops**, que permiten almacenar información a manera de bits. La cantidad de bits del registro queda determinada por la cantidad de biestables o flip-flops que posea. Los registros se clasifican según cómo se carguen los datos a la entrada y a la salida. Por un lado, están los que poseen carga paralela, tanto a la entrada como a la salida. Por otro lado, aquellos en que al menos la entrada o la salida es de carga serie. Estos últimos se denominan registros de desplazamiento.

REGISTROS DE ENTRADA Y SALIDA PARALELO

Una forma muy sencilla de obtener un registro de entrada y salida **paralelo** es empleando **flip-flops D** asincrónicos (latches) con entrada de habilitación (LE), que se unirán de modo tal de quedar una entrada de habilitación general. La cantidad de biestables que utilicemos determinará la cantidad de bits que almacenará el registro. El funcionamiento es de

La cantidad de bits del registro queda determinada por la cantidad de biestables o flip-flops que posea

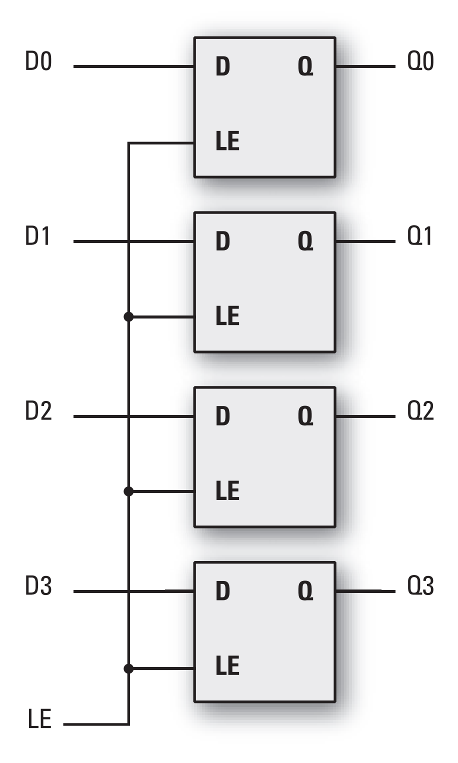


FIGURA 19. Vemos un registro de entrada y salida paralelo de 4 bits. La entrada de habilitación general permite mantener los estados de la salida.

la siguiente manera: cuando la entrada de habilitación general posee nivel **lógico 1** (habilitado), se habilitarán todos los biestables. Así, el bit que se encuentre en cada entrada **D** será cargado en el biestable correspondiente (**Figura 19**).

Al deshabilitar la entrada de habilitación general, la salida **Q** de cada biestable mantendrá el estado correspondiente, sin importar que se modifique el estado de la entrada **D**. En caso de que necesitemos colocar una habilitación para mostrar los datos a la salida, podemos hacerlo fácilmente agregando una compuerta **AND** de dos entradas para cada una de

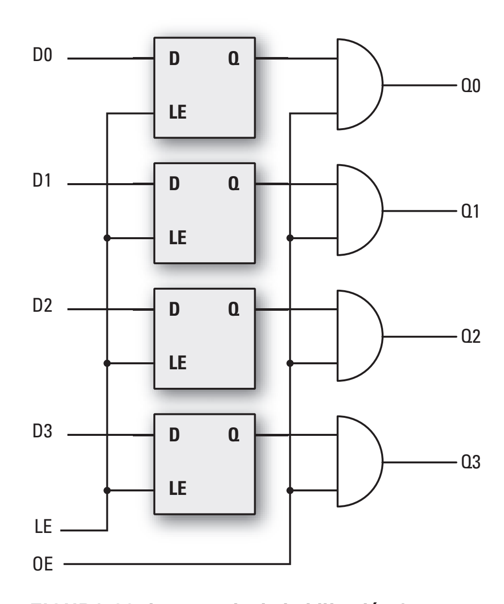


FIGURA 20. La entrada de habilitación Output Enable (OE) permite mostrar los datos a la salida solo cuando esté habilitada dicha entrada.

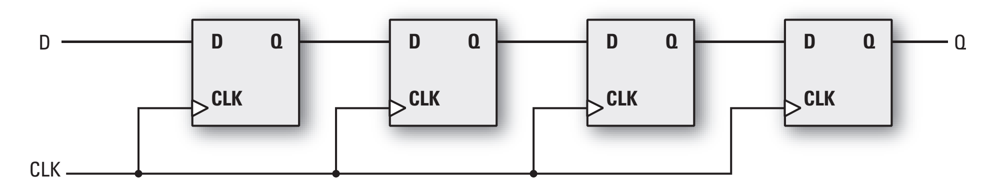


FIGURA 21. Observamos un registro de desplazamiento unidireccional, con entrada y salida serie.

las salidas. Una de las entradas de cada compuerta **AND** será la salida **Q** del flip-flop **D** correspondiente. La segunda entrada de cada **AND** será la misma para todas, y corresponderá a la entrada **Output Enable** (OE), ver (**Figura 20**).

REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento deben su nombre a que cada bit de dato se irá desplazando o corriendo en cada ciclo de la señal de reloj. Uno de los más sencillos corresponde al registro de desplazamiento con entrada y salida serie. Está compuesto por **n flip-flops D**, donde **n** será la cantidad de bits que almacene el registro. La señal de reloj es única y se conecta a cada entrada de reloj de los flip-flops. Luego, la conexión de los flip-flops es en cadena, es decir, la salida **Q** del primero se conectará a la entrada **D** del segundo; luego la salida **Q** del segundo se conectará a la entrada **D** del tercero, y así sucesivamente. La entrada **D** del primer **flip-flop** es la entrada serie del registro. Los bits ingresados por ella se irán desplazando en cada ciclo de la señal de reloj por cada uno de los flip-flops, hasta llegar al último de ellos, siendo la salida **Q** de éste, la salida serie del registro (**Figura 22**).

Existen registros de desplazamiento **unidireccionales**, que permiten que los bits de desplacen en una única dirección (**Figura 21**). También están los **bidireccionales**, que permiten que los bits se desplacen en ambos sentidos.

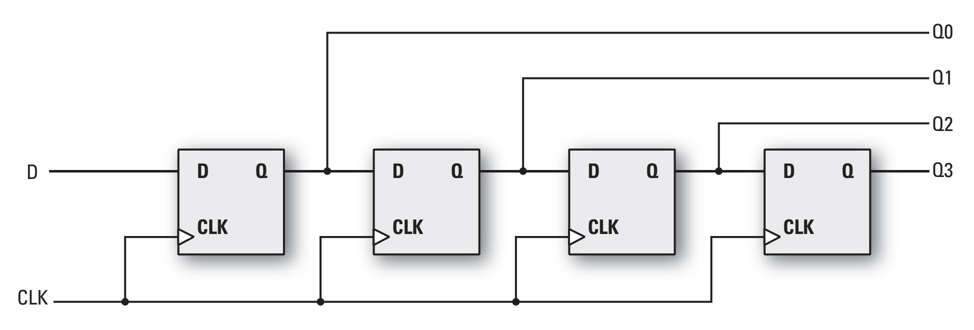


FIGURA 22. El mismo registro de desplazamiento con entrada y salida serie puede ser de entrada serie y salida paralelo, tomando los bits de cada salida Q de los flip-flops.

Existen registros de desplazamiento unidireccionales y bidireccionales, según el sentido de movimiento

Si en vez de tomar solo la salida **Q** del último flip-flop, tomamos la salida **Q** de cada uno, vemos que el mismo registro de desplazamiento con entrada y salida serie se convierte en un registro de desplazamiento con entrada serie y salida paralelo (**Figura 22**).

Existen numerosos dispositivos que requieren de registros para su funcionamiento. Más adelante veremos usos prácticos de estos elementos

Registros de **desplazamiento**

Ya sabemos cómo funcionan los registros de desplazamiento. Ahora veremos de qué manera utilizarlos para construir un sencillo circuito simulador de vela u hogar a leña.

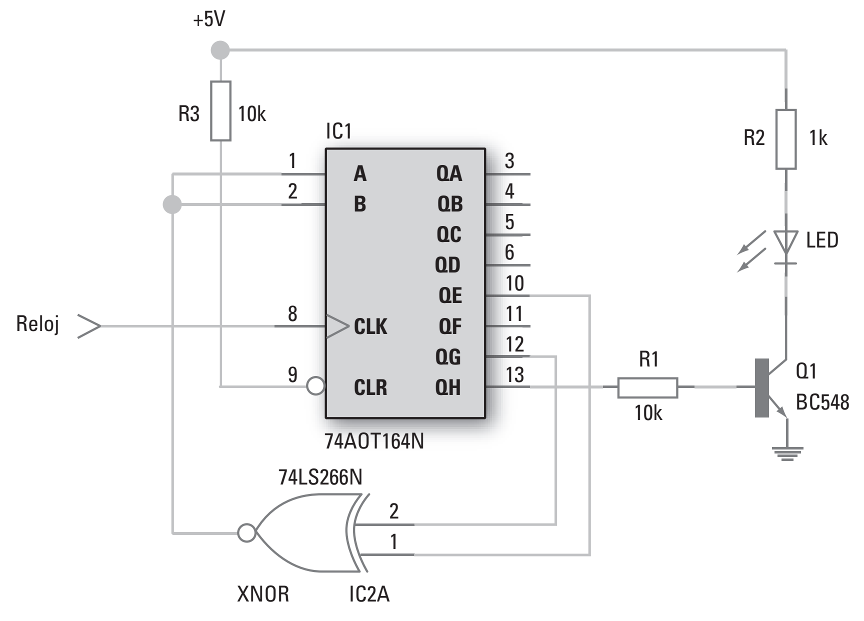
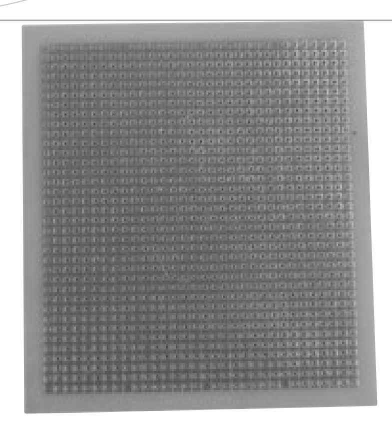


FIGURA 23. Esquema del circuito simulador de vela/hogar a leña. La base del circuito es un registro de desplazamiento 74164, pero podríamos utilizar cualquier otro modelo siempre y cuando sea entrada serie/salida paralelo.



En la **Figura 23** observamos un circuito muy similar al utilizado para generar y verificar el **CRC** (*Cyclic Redundancy Check*) con el que se detectan errores en sistemas de comunicaciones, como Ethernet.

La base del circuito es un registro de desplazamiento de entrada serie/salida paralelo. La señal de reloj debe encontrarse en el orden de los **KHz** y podemos generarla mediante un **circuito integrado 555** operando en modo estable o con un par de compuertas lógicas. El desplazamiento de la información en el registro se produce con cada pulso de reloj. La compuerta **XNOR** toma las salidas **QE** y **QH** e introduce nuevos datos en la secuencia (realimentación).

TABLA	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

TABLA 9. Multiplexor que posee un valor de salida que corresponderá al valor de la entrada.

El dato introducido depende de los anteriores, y esto genera una secuencia pseudoaleatoria, cuyo tiempo de repetición depende de las salidas consideradas y del módulo del contador

Las salidas generalmente no son capaces de suministrar la corriente necesaria para encender un LED. Es por esta razón que se utiliza un transistor como el **BC548**, operando como llave.

Como la salida de **QH** es una secuencia pseudoaleatoria, el LED cambia su brillo de manera aleatoria, para dar un bonito efecto de luz de vela. La resistencia **R3** de **10 K** se utiliza para proporcionar una referencia que asegura el **1** lógico en la entrada de **CLR**, manteniéndola deshabilitada.

Multiplexores

A continuación, explicaremos qué son y para qué sirven los multiplexores, y veremos también la forma de realizar expansiones con estos dispositivos. Los multiplexores son dispositivos que nos permiten mostrar una única señal en su salida, entre varias señales de entrada. La selección se realiza mediante una señal de control.

TEORÍA DE FUNCIONAMIENTO

En electrónica digital, los multiplexores son dispositivos que poseen **2n** entradas (**I**), una sola salida (**Y**) y **n** entradas de control (**S**). Así, por ejemplo, un multiplexor de cuatro canales (**4** entradas) tendrá dos entradas de control (**S0** y **S1**). El valor de la salida corresponderá al valor de la entrada seleccionada mediante las entradas de control. En la **Tabla 9** podemos ver el ejemplo mencionado.

EXPANSIÓN

La expansión de multiplexores nos permite obtener un nuevo multiplexor con mayor cantidad de canales. Esto se logra mediante un esquema de expansión en árbol (**Figura 24**). Supongamos que contamos con cinco multiplexores (M1, M2, M3, M4 y M5) de cuatro canales cada uno. Con ellos, podremos conseguir un multiplexor de 16 canales con sus cuatro entradas de control correspondientes, de la siguiente forma: conectamos las salidas **Y** de M1, M2, M3 y M4 a las entradas **I0**, I1, I2 e I3 de M5. Las entradas **I** de M1, M2, M3 y M4 (16 en total) serán las nuevas entradas **I**. La salida **Y** de M5 será la nueva salida **Y**. La nueva señal de control será de 4 bits, donde las dos entradas de control de M5 serán los dos bits más significativos (**S3** y **S2**), y las entradas de control del resto corresponderán a los dos bits menos significativos (**S1** y **S0**).

Los multiplexores son dispositivos que poseen 2n entradas, una sola salida y n entradas de control

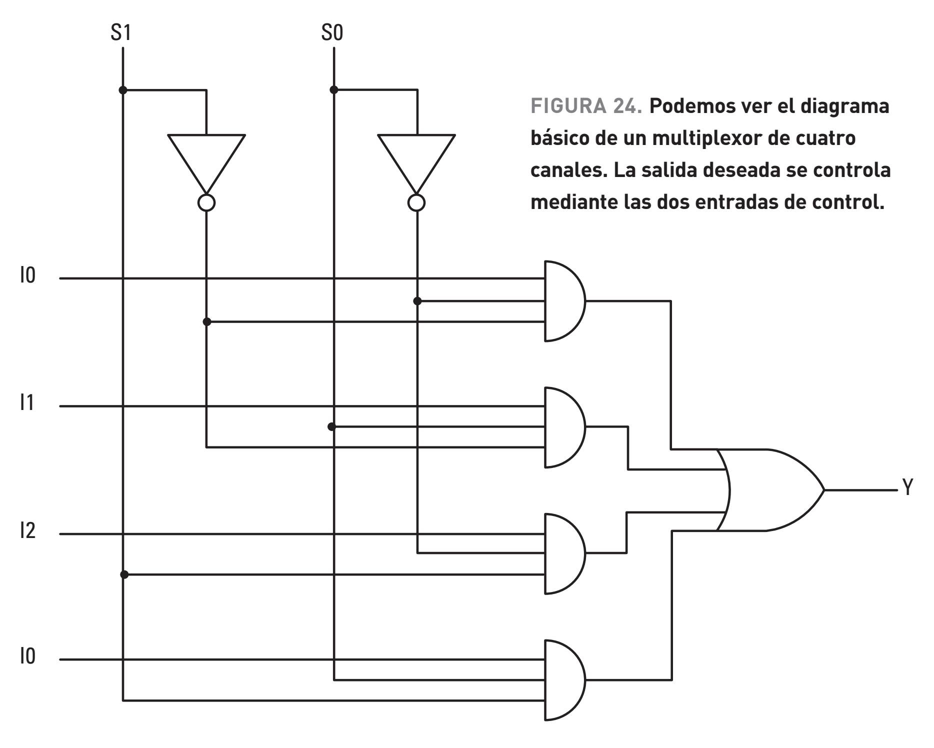


FIGURA 24. Podemos ver el diagrama básico de un multiplexor de cuatro canales. La salida deseada se controla mediante las dos entradas de control.

Demultiplexores

El funcionamiento de los **demultiplexores** es exactamente **inverso** al de los multiplexores. Son dispositivos que poseen una sola entrada, **2n** salidas y **n** entradas de control. De este modo, mediante las entradas de control, seleccionamos la salida por la cual se transferirá la señal que posee a la entrada **(Figura 25**).

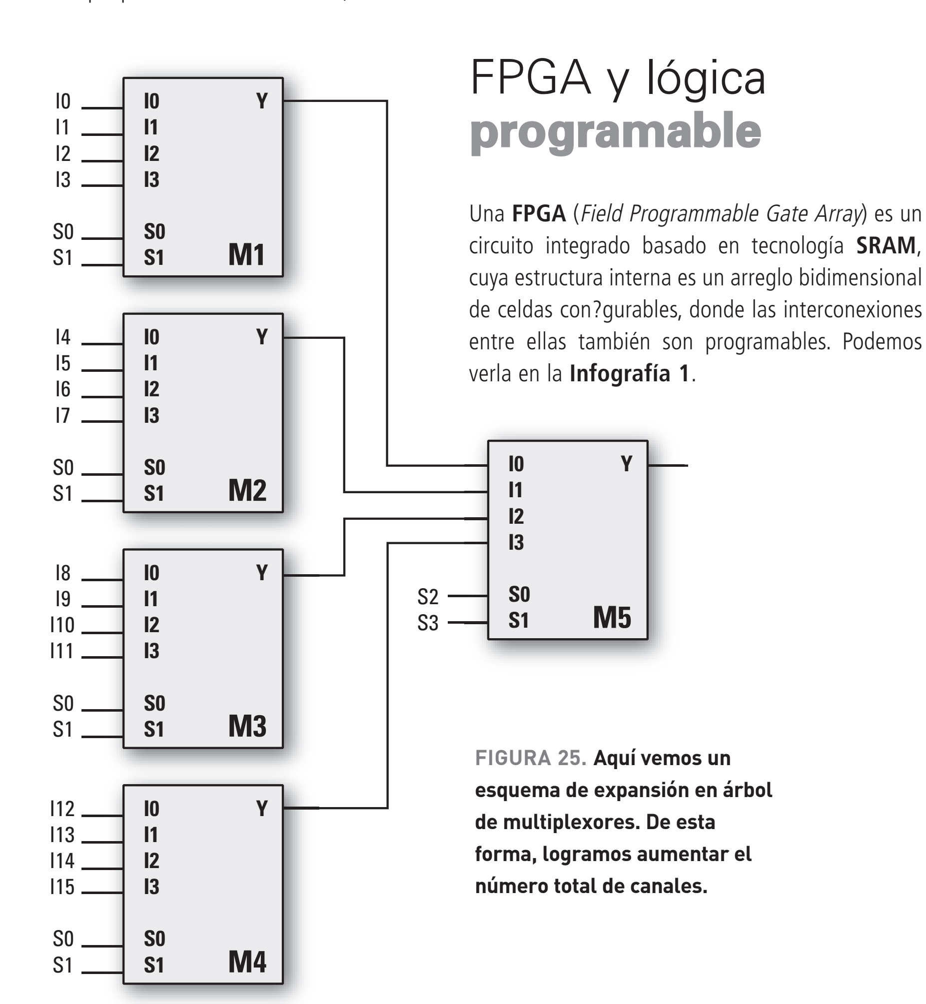


FIGURA 25. Aquí vemos un esquema de expansión en árbol de multiplexores. De esta forma, logramos aumentar el número total de canales.

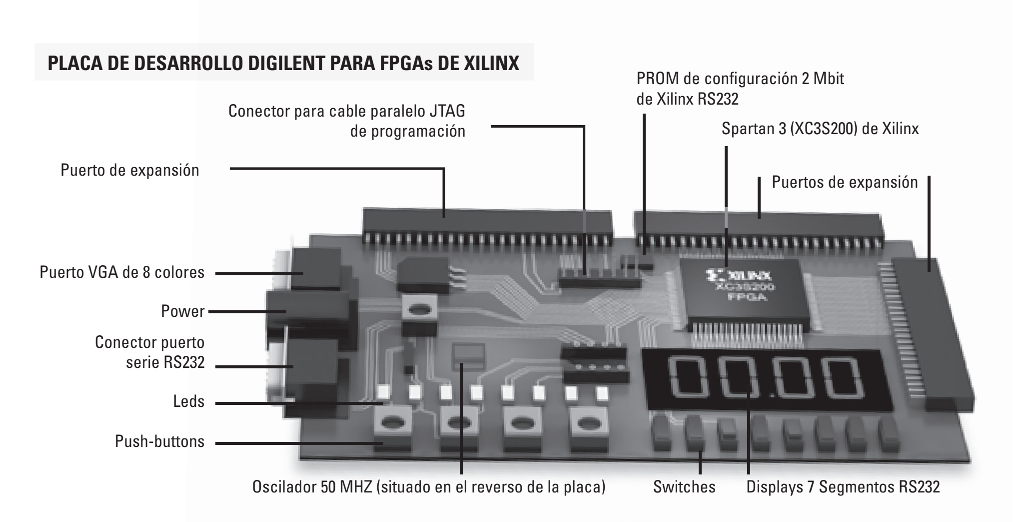
FPGA y lógica **programable**

Una **FPGA** (*Field Programmable Gate Array*) es un circuito integrado basado en tecnología **SRAM**, cuya estructura interna es un arreglo bidimensional de celdas con?gurables, donde las interconexiones entre ellas también son programables. Podemos verla en la **Infografía 1**.



INFOGRAFÍA 1: FPGA Y LÓGICA PROGRAMABLE

PLACA DE DESARROLLO DIGILENT PARA FPGAs DE XILINX



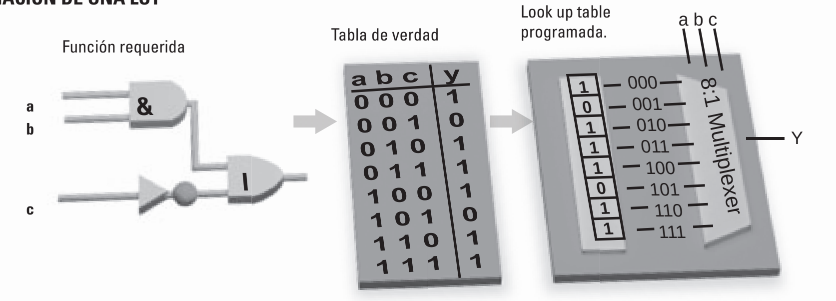
MODELO SPARTAN 3

Device	Número de celdas lógicas	Número de Block RAMs	Número digital de clock managers
XC3S50	1728	4	2
XC3S200	4320	12	4
XC3S50	8064	16	4

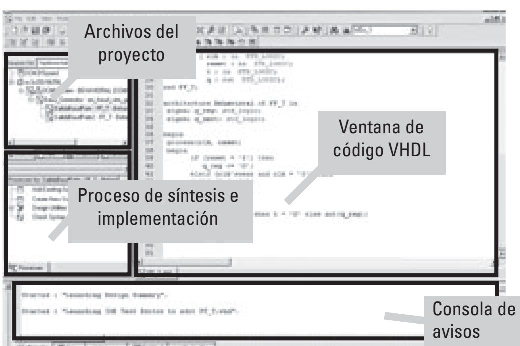
TIP

CPLDs (Complex Programmable Logic Device): Son dispositivos programables, pero en este caso, basados en celdas ROM (eprom, e2prom y flash) de menor capacidad lógica y performance de velocidad que una FPGA. Generalmente usados como "Glue Logic" (circuitos de interfaz entre integrados, y entre integrados y salidas).

PROGRAMACIÓN DE UNA LUT



SOFTWARE DE DISEÑO



En el caso de Xilinx, ISE Webpack es el software gratuito para especificación, síntesis e implementación de sistemas digitales en FPGAs y CPLDs. Los dos lenguajes más utilizados para la especificación son VHDL y Verilog. Utilizan una semántica de descripción de "procesos"de ejecución paralela.

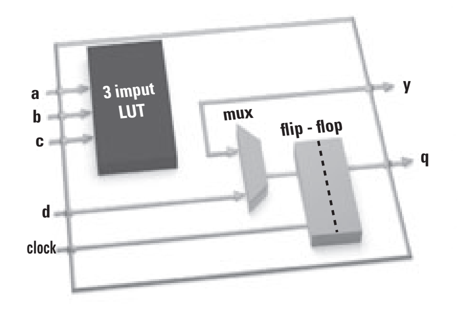




CELDA LÓGICA

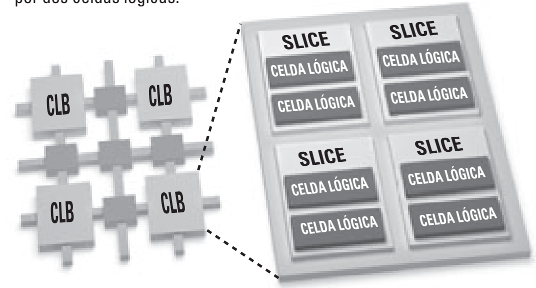
Una celda lógica consta de un pequeño circuito combinacional configurable y un flip-flop tipo D. El circuito combinacional puede ser implementado por una LUT (Look Up Table), que a los fines prácticos puede ser considerada una memoria de capacidad 2^N por 1.

Grabando el contenido de esta memoria, es posible implementar en la LUT cualquier función de N entradas. El flip-flop es para uso en circuitos secuenciales.



BLOQUE LÓGICO CONFIGURABLE (CLB)

Un CLB está compuesto por 4 slices y cada slice está compuesto por dos celdas lógicas.



Multiple **choice**

1 **¿Cuántas entradas tiene el circuito secuencial flip-flop RS?**

a- 2.
b- 4.
c- 6.
d- 16.

---

2 **¿Cuál de las siguientes no es una clasificación válida para un contador?**

a- Binarios/no binarios.
b- Síncronos/sincrónicos.
c- Asíncronos/asíncrónicos.
d- Decimal/Adecimal.

---

3 **¿Cuántos flip-flops JK conectados en serie posee el contador binario asíncrono de 4 bits?**

a- 2.
b- 4.
c- 6.
d- 16.

---

4 **¿Cuántos estados diferentes posee el contador binario asíncrono de 4 bits?**

a- 2.
b- 4.
c- 6.
d- 16.

---

5 **¿En qué tipo de flip-flops se basa el contador en anillo?**

a- A.
b- B.
c- C.
d- D.

---

6 **¿Por qué tipos de flip-flops está compuesto el registro de desplazamiento?**

a- A.
b- B.
c- C.
d- D.

Respuestas: 1 a, 2 d, 3 b, 4 d, 5 d, 6 d.

Capítulo 3

Memorias



Analizaremos las características de los medios de almacenamiento y procesamiento de la información.



Memorias

Conoceremos las características de los principales medios de almacenamiento y procesamiento de la información en un sistema microcontrolado. Abordaremos los conceptos teóricos relacionados con el almacenamiento digital de datos. No entraremos en detalle en los medios masivos, sino que nos enfocaremos en aquellos que se utilizan mayormente en sistemas basados en microcontroladores, ya que es con estos que haremos la mayoría de nuestras experiencias a lo largo de este libro.

La variedad existente en tipos de memoria depende en gran medida de las particularidades que cada uno aporta. Cada aplicación tiene sus requerimientos específicos, que se benefician de una u otra particularidad y, los microcontroladores que veremos en adelante no son la excepción.



FIGURA 1. El disco rígido permite guardar grandes cantidades de información utilizando principios magnéticos.

La variedad existente en tipos de memoria depende en gran medida de las particularidades que cada uno aporta

Almacenamiento digital

La necesidad de guardar información de manera confiable y perdurable llevó a diseñar gran cantidad de dispositivos de almacenamiento, entre los cuales se encuentran las memorias electrónicas.

En la actualidad utilizamos el concepto de almacenamiento digital para denominar a la capacidad de guardar información de variados orígenes en formato digital, es decir, convertida en datos binarios (unos y ceros). El uso de esta moderna tecnología permite archivar, gestionar, buscar y compartir información de manera electrónica utilizando distintos medios de soporte.

Los medios físicos de almacenamiento digital pueden ser de diferentes tipos, de acuerdo a la naturaleza física del sistema de almacenamiento (**Figuras 1** y **2**). Esto es: **magnéticos** (disquetes, cintas y discos rígidos), **ópticos** (MiniDisc, CD, DVD, Blu-ray) y **electrónicos** (memorias de tecnología semiconductoras, por ejemplo **ROM**, **PROM**, **EPROM**, **EEPROM**, **EEPROM Flash**, **RAM**, **DRAM**, **SRAM**, **SDRAM**, **RDRAM**, **FRAM**).



FIGURA 2. Memoria MBM27256-20 **de tipo** EPROM nMOS **de** 32 k x 8 bits **construida por la empresa** Fujitsu. **La ventana que posee permite borrar su contenido mediante luz** UV.

VENTAJAS DEL ALMACENAMIENTO DIGITAL

El almacenamiento digital de la información trae muchas ventajas consigo, ya que no solo es posible guardar información, sino que también podemos recuperarla, procesarla o modificarla de una manera rápida y fácil. Esta tecnología permite un gran ahorro del espacio físico utilizado para el almacenamiento de los medios y una disminución del riesgo de la pérdida de información. Los documentos almacenados se mantienen inalterables a través del tiempo, ya que no se producen deterioros físicos. Otra ventaja es la duplicación fiel que se puede realizar de todos los documentos almacenados en cuestión de segundos. Esto es muy importante a la hora de realizar copias de respaldo para asegurar la perpetuidad de los contenidos.

Los dispositivos de almacenamiento ópticos están construidos alrededor de discos de material plástico

MEDIOS MAGNÉTICOS

Este tipo de memorias utilizan un soporte físico, como ser un plato o un bobinado de cinta, el cual se encuentra recubierto por un material magnético. A su vez, la superficie magnética se encuentra dividida en gran cantidad de regiones pequeñas que al ser polarizadas de manera adecuada permite almacenar los valores binarios **1** o **0**. Luego, con un sensor adecuado es posible obtener la lectura de esos datos magnéticos para luego convertirlos en pulsos eléctricos que permiten recuperar la información. Generalmente se trata de dispositivos lentos para su acceso pero de gran capacidad, muy utilizados en sistemas de respaldo.

MEDIOS ÓPTICOS

Los dispositivos de almacenamiento ópticos están construidos alrededor de discos de material plástico, en el cual, sobre una de sus caras, se graban de manera microscópica pequeños surcos que representan secuencias de datos binarios. La otra cara posee una cubierta metálica, generalmente de aluminio. El sensor empleado para recuperar la información es un haz láser que explora la superficie, rebota en la cubierta metálica y regresa excitando a un arreglo de fotodiodos que convierten la luz en variaciones eléctricas. Los soportes más populares de este tipo son los CDs, DVDs y Blu-ray. (**Figura 3**).

MEDIOS ELECTRÓNICOS

En un sistema electrónico se considera como memoria a cualquier tipo de dispositivo físico que posea la capacidad de almacenar información. En este caso se trata de componentes electrónicos basados en materiales semiconductores como diodos o transistores. Debido a que las memorias electrónicas trabajan de manera digital, la única forma de enviarles datos es en dicho formato. En el caso de que el material para almacenar sea una señal de origen analógica, como una grabación de sonido, será necesario realizar una conversión analógica a digital para luego sí poder trabajar con los datos.

Las memorias almacenan **datos** (1 o 0) organizados en un conjunto de bits llamado **palabra de datos** o **registro,** que se transfiere de forma simultánea desde y hacia la memoria. Cada palabra de datos se ubica en una localización específica, que recibe el nombre de **Dirección**.

Las memorias transfieren datos en forma de palabra binaria de 8, 16, 32 e incluso de hasta 64 bits

CELDA DE MEMORIA

La unidad mínima de almacenamiento de una memoria es denominada **celda** y permite guardar un único dato que puede adoptar solamente dos valores 1 o 0. Las celdas, en su forma genérica, están construidas alrededor de un **capacitor** o un **flip-flop**. En el caso del capacitor, este almacena el valor **1,** cuando se encuentra cargado eléctricamente, y el valor **0**, cuando está descargado.



FIGURA 3. Disco de estado sólido. Lo formatos de cinta magnéticas se encuentran en desuso al ser reemplazadas por medios electrónicos, como los discos de estado sólido.

Las celdas construidas alrededor de **flip-flops** almacenan el dato **1** cuando el flip-flop presenta un nivel alto en la salida **Q**.

Además de la celda, una memoria en su interior emplea un sistema de decodificación para direccionar los datos de entrada y de salida hacia cada celda de forma lineal. Gracias al decodificador es posible reducir el número de líneas de acceso a la memoria, ya que a partir de **n** cantidad de líneas de entrada obtenemos **2^n** (2 elevado a la N) líneas de salida.
Por ejemplo, una memoria con una capacidad de almacenamiento de **128 bits** internamente se encuentra organizada en **32** direcciones con una capacidad de **4 bits** cada una. Para direccionar esta cantidad de datos, nuestro decodificador necesitará solamente **5 líneas** de acceso, debido a que a su salida entregará **2^5 = 32** direcciones posibles (**Figura 4**). Generalmente, la capacidad de una memoria se expresa como el resultado de multiplicar el número de posiciones que posee por la longitud de cada palabra de datos.

La unidad mínima de almacenamiento de una memoria es denominada Celda y permite almacenar un único dato

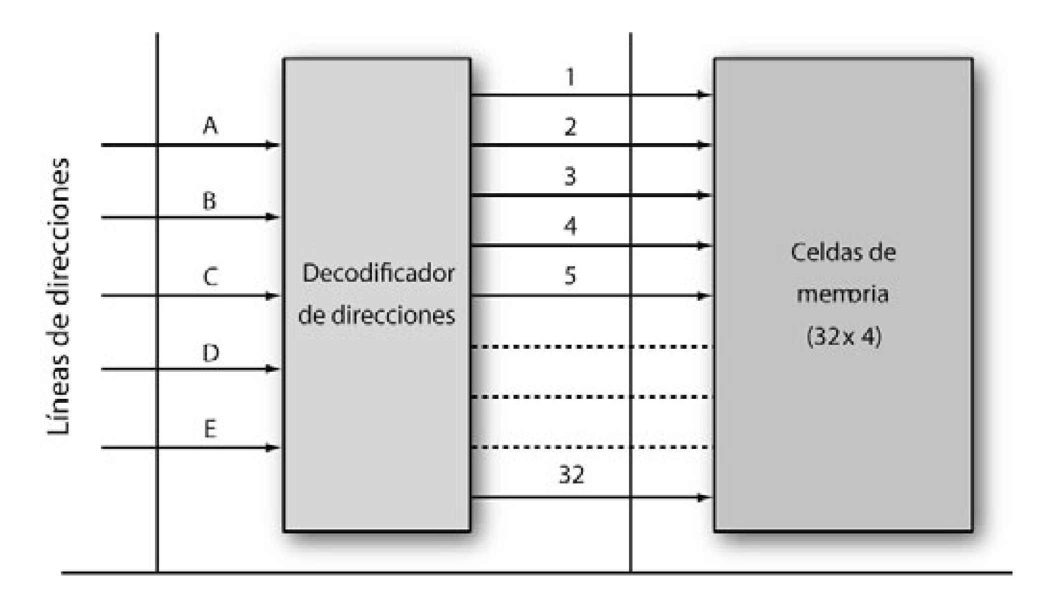


FIGURA 4. Estructura interna de una memoria con una capacidad de 128 bits. Dispone un total de 32 direcciones con capacidad para palabras de 4 bits y tan solo 5 líneas de acceso.

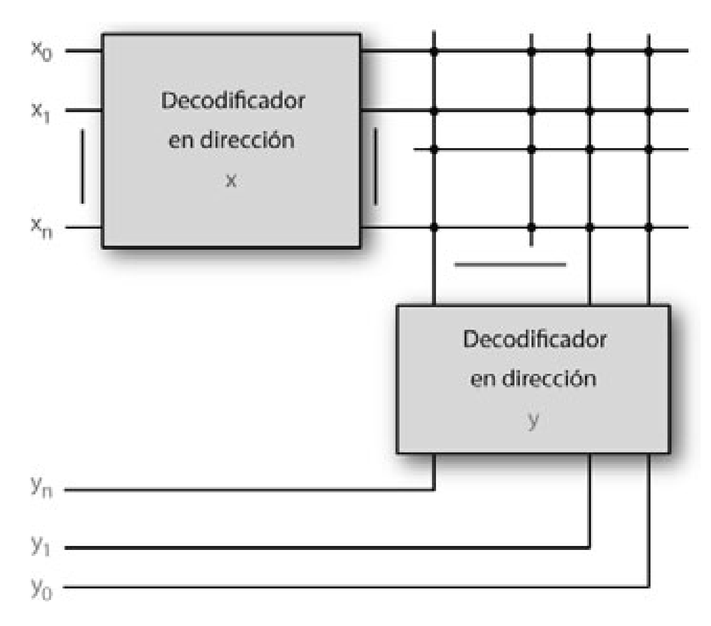


FIGURA 5. **Organización matricial de una memoria. Es posible acceder a cada celda de almacenamiento a partir de dos decodificadores de direccionamiento** X **e** Y**.**

ORGANIZACIÓN MATRICIAL

Cuando la capacidad de la memoria es elevada, se torna difícil acceder de forma lineal a su contenido. Es por ello que surge la estructura matricial para organizar internamente al dispositivo. Este tipo de arreglo dispone de celdas situadas en filas y columnas. Las líneas de direccionamiento se encuentran divididas en dos bloques: **horizontal** (**X**) y **vertical** (**Y**). Al utilizar esta estructura, es posible activar las celdas de memoria de manera individual e, incluso, de forma aleatoria. Esta es la principal característica de las llamadas memorias **RAM** (Memoria de Acceso Aleatorio, del inglés *Random Access Memory*).

LECTURA DE UNA MEMORIA

Para realizar una lectura del contenido de una memoria es necesario seguir una serie de pasos: debemos colocar el número binario que represente la dirección a la cual queremos acceder en las líneas de direccionamiento. Luego, será necesario habilitar el comando de lectura. Generalmente, las memorias disponen de un pin dedicado que permite escoger el modo de funcionamiento entre lectura y escritura. Al leer una memoria surge un parámetro conocido como **tiempo de acceso**. Dicho parámetro mide el tiempo transcurrido entre que se recibe una dirección valida, se habilita el modo de lectura y aparecen los datos en las líneas de salida.

ESCRITURA DE UNA MEMORIA

Para poder escribir datos en una memoria será necesario seguir varios pasos: mediante las líneas de direccionamiento, debemos elegir la dirección de la posición donde almacenaremos los datos. A continuación, tenemos que colocar los datos en las líneas de entrada. Por último, debemos habilitar el pin de **escritura/lectura** en el modo correspondiente.

Cada vez que escribimos en la memoria, debemos respetar un parámetro conocido como **tiempo de ciclo**. Este parámetro indica el tiempo que tienen que estar presentes los datos a escribir y los datos de direccionamiento mientras dure el proceso (**Figura 5**).

MEDIDAS DE ALMACENAMIENTO DIGITAL

Los medios de almacenamiento digital disponen de una capacidad para guardar datos, dicha capacidad puede ser medida en bits y sus respectivas derivaciones.

bit	b	Un dígito binario.
byte	B	Submúltiplo del tamaño de una palabra, generalmente de ocho bits.
octeto	o	Grupo de ocho bits.

TABLA 1. Se muestran las unidades para representar cantidades en electrónica digital.

Un **bit** representa a un dígito del sistema de numeración binario. En dicho sistema se utilizan solo dos dígitos, el **0** y el **1**, a diferencia del sistema decimal, en el cual se utilizan diez dígitos. Un bit puede adoptar uno u otro valor exclusivamente (**Tabla 1**).

Como el bit es una unidad de almacenamiento muy pequeña para los tiempos que corren, generalmente se utiliza asociado con lo que se denomina prefijos de cantidad. Según el estándar **IEEE 1541** los tradicionales prefijos del **SI** (Sistema Internacional) no se pueden utilizar para representar múltiplos binarios. Por lo tanto, todas las unidades utilizadas para indicar cantidades de almacenamiento en el campo de la electrónica digital y la informática poseen una nueva denominación: se introduce la parte **bi** (de binario) al prefijo. Por ejemplo, lo que antiguamente era un **kilobyte** ahora es equivalente a un **kibibyte**, que significa un **kilobinario byte**, que son **1.024 bytes** (**Tabla 2**).

kibi	Ki	210 = 1.024
mebi	Mi	220 = 1.048.576
gibi	Gi	230 = 1.073.741.824
tebi	Ti	240 = 1.099.511.627.776
pebi	Pi	250 = 1.125.899.906.842.624
exbi	Ei	260 = 1.152.921.504.606.846.976

TABLA 2. Prefijo para indicar los múltiplos binarios de las unidades de la Tabla 1.



PINES DE CONTROL

Las memorias disponen de varios pines especiales como por ejemplo el pin de lectura/escritura que es el encargado de configurar el modo de funcionamiento. También es frecuente encontrar un pin de habilitación de dispositivo, Chip Select (CS).

Clasificación de los sistemas **de almacenamiento**

Los sistemas de almacenamiento se clasifican según varios aspectos. A continuación describiremos uno a uno los **principales grupos**.

SEGÚN SU MÉTODO DE ACCESO

Según el modo de acceso a los datos almacenados en un dispositivo de memoria podemos clasificarlos en dos grupos: **aleatorio** y **secuencial**. En el primero, se puede acceder a cualquier dato, sin importar su posición física, y con el mismo tiempo de acceso para todas las direcciones. Todas las memorias semiconductoras, como las **RAM** o **ROM**, pertenecen a este grupo. El segundo grupo, en cambio, obliga a leer o escribir todas las posiciones físicas previas antes de acceder al dato deseado. En este caso el tiempo de acceso cambia de acuerdo al sector físico al que debamos acceder.

SEGÚN SU VOLATILIDAD

De acuerdo a la capacidad de mantener los datos almacenados, podemos encuadrarlos en los siguientes grupos:

Las memorias volátiles pierden los datos al nterrumpir la energía que las alimenta

Por un lado, encontramos los **No volátiles,** donde la información es mantenida en la memoria a pesar de que el dispositivo no cuente con energía de alimentación. Teóricamente, el tiempo de almacenamiento es indefinido. Todos los medios del tipo magnético y óptico pertenecen a este grupo (**Figura 6**).

Por el otro lado están los **Volátiles**, estos pierden los datos al interrumpirse la energía que los alimenta. Estas memorias son llamadas de trabajo o temporales. Por ejemplo, podemos citar a todos los tipos de memorias **RAM** utilizadas en las computadoras (**Figura 7**).



FIGURA 6. **Las memorias EEPROM FLASH son una variante de la memoria EEPROM que permite escribir o borrar múltiples posiciones de memoria simultáneamente.**

SEGÚN SU MÉTODO DE ESCRITURA

Bajo esta denominación encontramos dos grupos de memorias: **de lectura y de escritura**. Las más utilizadas son las **RAM**, tanto del tipo **estáticas** (**SRAM**) como las **dinámicas** (**DRAM**). Una memo-



FIGURA 7. La memoria RDRAM fue creada para competir con la SDR SDRAM, pero terminó siendo la alternativa a la DDR SDRAM. Fue utilizada utilizó en la consola de Nintendo 64.

ria no volátil es el BIOS de una PC, la cual almacena la información básica para iniciar el sistema.

Las memorias **RAM dinámicas** están construidas sobre diminutos capacitores a modo de celdas. Debido al empleo de esta tecnología es necesario utilizar una circuitería de refresco para mantener cargados los capacitores y no perder los datos almacenados. En cambio, las memorias **RAM estáticas** están basadas en **flip-flops**, los cuales no necesitan ningún tipo de refresco para trabajar.

MEMORIA DE SOLO LECTURA

En este tipo de memoria, el acceso a los datos es rápido, pero el proceso de escritura es todo lo contrario. Son dispositivos diseñados para mantener la información sin cambios. Dentro de este grupo encontramos una gran variedad de memorias:

- **ROM (Read Only Memory):** memorias de solo lectura programadas por máscara, los datos almacenados se escriben en fábrica y no es posible modificarlos ni borrarlos (**Figura 8**).
- **PROM (Programmable ROM)**: memoria **ROM** programable una sola vez. Al realizar el proceso de grabación el chip queda inalterable.
- **EPROM (Erasable PROM)**: memoria que puede borrarse mediante luz ultravioleta para regrabarlas.
- **OPT (One Time Programmable)**: son memorias **EPROM** sin ventana, por lo tanto no pueden borrarse.
- **EEPROM (Electrically Erasable PROM)**: memorias que pueden borrarse y regrabarse eléctricamente.
- **FLASH**: es una variación de las memorias **EEPROM**. Son más veloces que estas, de mayor capacidad y menor consumo.



APLICACIONES DE LAS MEMORIAS ROM

Las memorias ROM generalmente se utilizan para contener pequeñas porciones de código, dedicado a manejar hardware a bajo nivel en sistemas microcontrolados, y para almacenar patrones aplicados en gran variedad de aplicaciones, como tablas de datos.

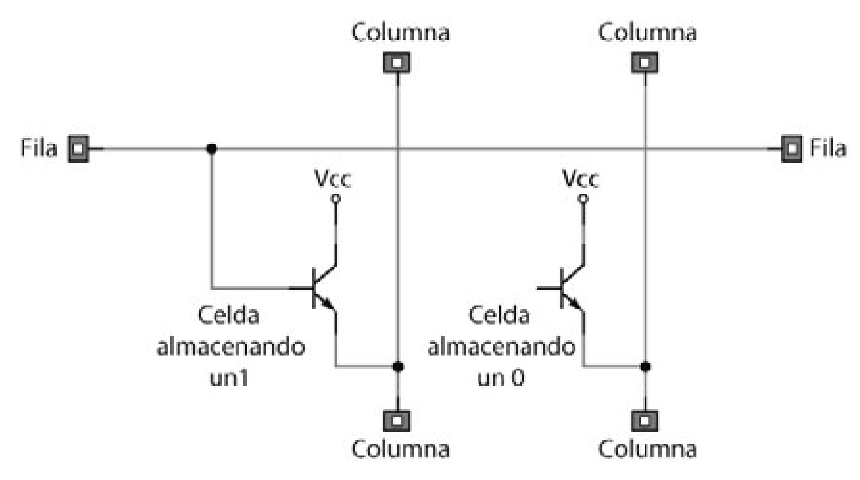


FIGURA 8. Celda de memoria de una ROM formada a partir de transistores bipolares. El conexionado se realiza en el proceso de fabricación de acuerdo a un patrón o mascara.

MEMORIAS ROM

Si realizamos un análisis funcional de la memoria ROM, podemos interpretarla como un **bloque de formato matricial** formado por tantas filas como de direcciones disponga y tantas columnas como la longitud de la palabra de datos.

Para acceder a este bloque se utiliza una combinación de números binarios que se aplican en la entrada de un circuito decodificador de **n líneas** de entradas y **2n** líneas de salida. En cada unión entre fila y columna se encuentran las celdas de memoria formadas por elementos semiconductores como pueden ser diodos, transistores bipolares o MOS. acuerdo al conexionado de dichos elementos la celda almacenará un **1** o un **0**.

Las celdas de una memoria PROM se construyen alrededor de fusibles, diodos y transistores

Por ejemplo, en una memoria con celdas construidas con transistores bipolares para que el contenido sea un 1, la base del transistor deberá conectarse a la línea de la fila. En cambio, si se desea almacenar un 0 la base quedará sin conexión.

MEMORIAS PROM/PROM CON DIODOS

Una memoria **PROM** es un tipo de **ROM** que puede programarse una única vez. Está construida con una especie de fusibles o antifusibles, que se queman de

manera irreversible haciendo circular por ellos una corriente excesiva. Los chips nuevos contienen todos los fusibles intactos, por lo cual, todo su contenido se encuentra en estado alto.

Las celdas de memoria, además del fusible, poseen cada una un elemento semiconductor asociado, en este caso un diodo, que es el encargado de definir el valor almacenado según se encuentre conectado o abierto.

Al realizar el proceso de grabación, se aplica una tensión elevada a la celda deseada que logra fundir el fusible, abriendo el circuito y dejando al diodo sin conexión. De esta forma la celda pasa a almacenar el valor **0**. El proceso de quemado se realiza con una herramienta llamada **programador de PROM**, la cual posee la capacidad de elegir una a una las direcciones de la memoria y asignarle el valor que queremos almacenar (**Figura 9**).

Aplicación de las **memorias PROM**

Una memoria PROM puede programarse para almacenar distintos códigos. En este ejemplo veremos la manera de utilizarla como un decodificador de BCD a 7 segmentos.

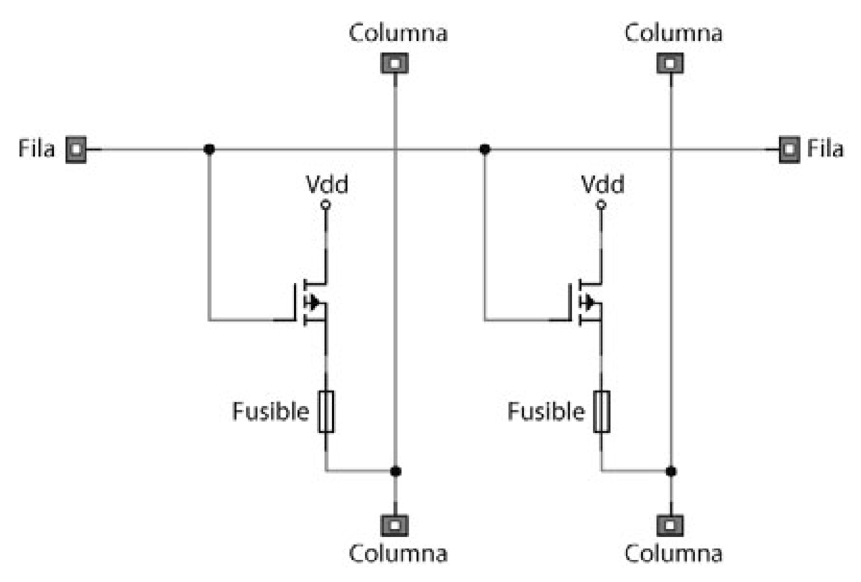


FIGURA 9. Celdas de una memoria PROM formada a partir de fusibles y transistores de tecnología MOS. Con este arreglo es posible almacenar una palabra de datos de 2 bits de longitud.

Un decodificador es un circuito digital de lógica combinacional que posee una **n** cantidad de pines de entrada y de salida. Cada combinación de datos en su entrada representa un único valor de salida. La función principal de estos dispositivos es la de realizar direccionamientos de espacios de memoria.

EL CIRCUITO PROPUESTO

Es posible almacenar una serie de datos en una memoria **PROM con diodos**, para que, a partir de una entrada de datos codificados en **BCD** (palabra de 4 dígitos binarios), se pueda realizar la correspondiente decodificación. De este modo es posible obtener a su salida **7 líneas** de datos para excitar un **display de 7 segmentos** y formar los dígitos decimales del **0** al **9**.

Nuestro circuito de ejemplo se encuentra formado por el bloque decodificador de direcciones (descrito al principio de esta página), una matriz con las celdas de memoria, una serie de diodos (en caso de querer almacenar el dato 1), un conjunto de resistencias **pull-down** (para mantener en nivel 0 cada celda que no posea conexión mediante diodo) y una serie de buffers a la salida. Estos últimos, son un tipo de compuerta con una alta capacidad de corriente que permite manejar directamente pequeñas cargas como un **LED**. Utilizaremos como dirección de los datos un número BCD y el dato de salida será el correspondiente al código de 7 segmentos.

La información a almacenar en nuestra memoria se toma de la correspondiente tabla de verdad (**Figura 10**). En nuestro caso, como solo aprovecharemos los datos correspondientes a los decimales del 0 al 9 (un total de 10 combinaciones), utilizaremos el código **BCD** desde **0000** a **1001** y el resto será ignorado.

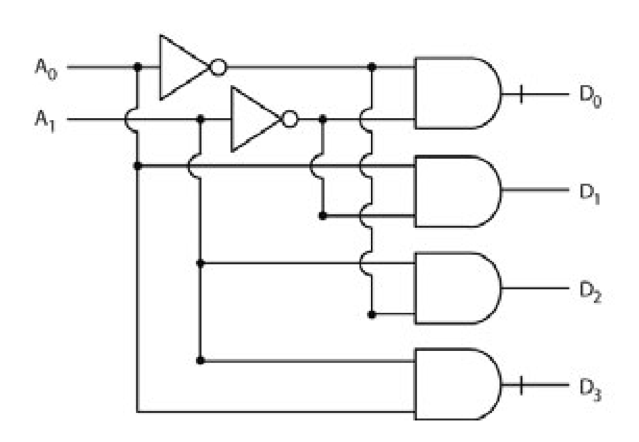


Tabla de verdad

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

FIGURA 10. Tabla de verdad de direccionamiento de datos de nuestro circuito. Los datos marcados con cruces son ignorados.

MEMORIAS EPROM

La memoria **EPROM** (*Erasable Programmable Read Only Memory* – **Memoria de Solo Lectura Programable y Borrable**) es una memoria no volátil que permite, como su nombre indica, ser programada y borrada por completo.

Su tecnología de fabricación permite que por medio de una fuente de luz ultravioleta todo su contenido sea borrado. Por este motivo, llevan en su encapsulado una pequeña ventana de cuarzo por la cual atravesarán los rayos ultravioleta. Para evitar que los datos almacenados en la **EPROM** sean alterados por fuentes lumínicas exteriores, especialmente el sol, se cubre la ventana con una etiqueta oscura que impide el paso de los rayos UV.

La programación de la esta memoria se diferencia de la **PROM** debido a que esta no requiere que sean fundidos sus fusibles, sino que induce cargas en la compuerta aislada de un transistor **MOS** por cada **celda/bit**. Cuando una **EPROM** está vacía o borrada por completo, todas sus celdas están descargadas. Por lo tanto, la salida será siempre un uno lógico (**1**) (el drenaje del transistor MOS). Dado que todas las celdas están en **1**, solo se programan los ceros, aunque deben introducirse todos los datos incluidos los unos (**1**). Este es el motivo por el cual es indispensable borrar por completo la memoria antes de ser reprogramada, ya que no es posible cambiar el valor de la celda de **0** a **1** (solo por UV), pero sí de **1** a **0** (**Figura 11**).

Una memoria vacía mostrará a su salida todos 1s, o lo que es lo mismo, en hexadecimal, todas FFs

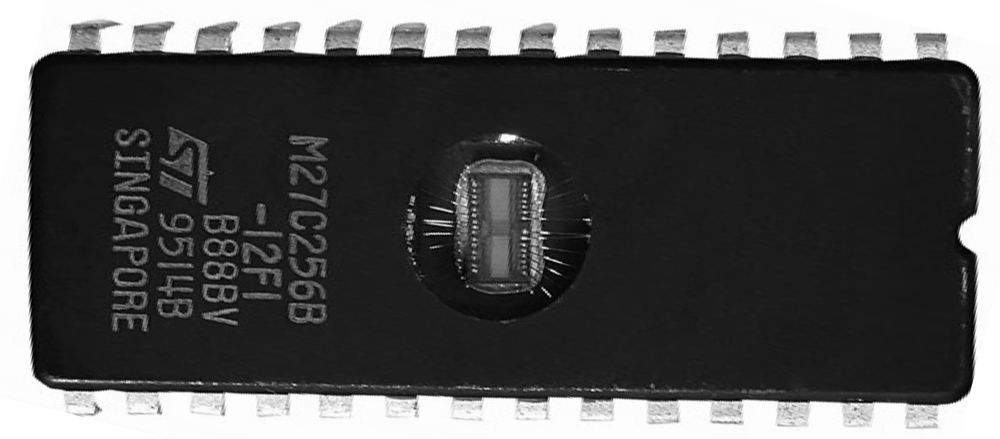


FIGURA 11. Memoria 27C256. **Podemos observar la ventana de cuarzo utilizada para borrar todo el contenido de la memoria por medio de rayos UV.**

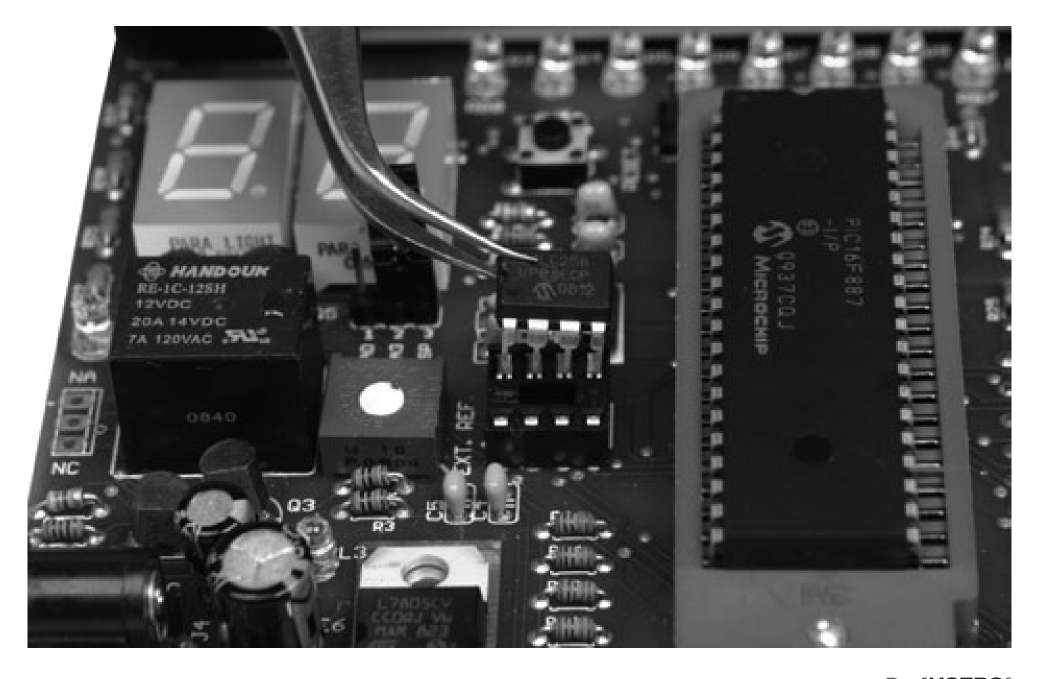
La retención de los datos en las memorias **EPROM** puede llegar a 20 años. Debido al costo de la ventana de cuarzo, se han implementado EPROMs grabables solo una vez (OTP – One Time Programmable) y su uso está destinado a grandes volúmenes de producción.

Únicamente están disponibles en formato de datos paralelo, esto es, cada bit de cada posición de la matriz de memoria está asociado a un pin del encapsulado. Por ejemplo, si una memoria tiene una palabra de **8 bits**, serán requeridos 8 pines para acceder a la palabra (o posición) seleccionada. Algunas memorias EPROM representativas son: 27C16, 27C32, 27C128, entre otras.

MEMORIAS EEPROM

La memoria EEPROM, también conocida como E2PROM, es idéntica a su antecesora EPROM, con la diferencia de que es una ROM Programable y Borrable Eléctricamente (Electrically Erasable Programmable Read-Only Memory). Esto significa que es posible borrar la memoria por medios eléctricos.

Por lo general las EEPROM pueden ser accedidas a través de un bus de datos paralelo o serie



Por lo general las **EEPROM** pueden ser accedidas a través de un bus de datos paralelo o serie, dependiendo del modelo. El formato serie permite utilizar encapsulados mas pequeños, ahorrando espacio y material en los **PCB**, además de ser más económicas.

La transmisión de datos en serie consiste en enviar por medio de un solo cable los **unos** y **ceros** uno detrás del otro. Si comparamos la transmisión serial con el acceso paralelo, notaremos que para obtener **8 bits** será necesario enviar (o recibir) uno a uno los datos, mientras que en **paralelo**, en una sola operación, obtenemos los datos de forma casi instantánea en los pines de la memoria. A pesar de la supuesta ventaja en la velocidad de las memorias en paralelo, las seriales son día a día más rápidas, superando en muchos casos a las memorias en paralelo. Hoy es casi imposible pensar en un desarrollo que requiera una memoria en paralelo. Veremos la comunicación en serie más adelante.

FIGURA 12. Debido al bajo costo de las memorias Flash su uso se ha masificado utilizándose en cientos de dispositivos.

Las memorias **EEPROM** pueden **borrarse/regrabarse** entre 100.000 y 1.000.000 de veces y, normalmente, el tiempo de almacenamiento es de **40 años**. Para borrar por completo o, sólo un sector de una memoria EEPROM, debe borrarse de a una posición por vez, hasta completar el proceso.

Algunas memorias **EEPROM** representativas son: 28C64, 28C128, 24C02, 93C86, 25C256.

MEMORIAS FLASH

Las memorias Flash son una evolución de las memorias **EEPROM**, y mantienen las mismas características de estas. Su principal diferencia es que pueden borrarse por completo en una sola operación, lo cual las hace mucho más veloces que las **EEPROM**, pero a veces esto mismo es un inconveniente. Desde hace algunos años, existen memorias Flash que permiten borrar por sectores llamados bloques, en lugar de borrar toda la matriz por completo. Los bloques son grandes sectores en los cuales está dividida toda la matriz de la memoria (**Figura 12**).



LECTURA Y ESCRITURA

En las memorias paralelas es común la siguiente secuencia para la grabación: Cargar la dirección. Configurar el pin R/W (escritura-lectura) . Para grabar, cargamos los datos en el bus de datos. Finalmente activamos y desactivamos el pin CS (Chip-Select).

Debido a que la grabación requiere de un voltaje superior al de alimentación, en la actualidad ese voltaje es generado internamente, liberando así al desarrollador de tener que incorporar una tensión extra al circuito. Estas memorias pueden soportar entre 100.000 y 1.000.000 de ciclos de borrado/reprogramación y 100 años de retención en los datos.

Las memorias Flash representativas son: 49F010, 29C512, 25FS010

MEMORIAS RAM

RAM significa **Memoria de Acceso Aleatorio** (*Random Access Memory*), es decir, podemos acceder a cualquier posición de la memoria para poder leerla o escribirla, (**Figura 13**). Ahora bien, una memoria semiconductora **ROM** y sus derivadas también permiten un acceso aleatorio. Entonces, ¿cuál es la diferencia, si en ambos tipos podemos acceder a cualquier posición y obtener nuestros datos? En los primeros tiempos de la computación, una de las memorias de almacenamiento de programas era la cinta magnética, y como es lógico suponer, el acceso era secuencial, es decir, para obtener un dato grabado al final de la cinta, debíamos pasar por todas las posiciones anteriores antes de dar con él. En cambio, en la memoria **RAM** (si bien no es usada para almacenar programas) se puede acceder directamente a cualquier posición independientemente de cuál sea la última posición leída. Esta es la historia de su denominación.

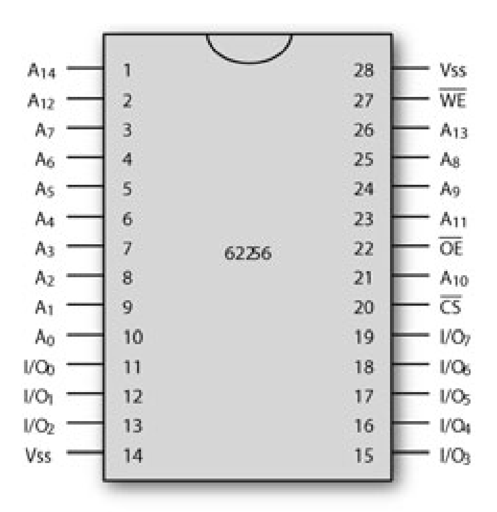


FIGURA 13. **Esta es la distribución de pines de una memoria RAM 62256 capaz de almacenar 32.768 palabras de 8 bits.**

Todas las memorias semiconductoras (sean estas **ROM** o **RAM**) son de acceso aleatorio. Por lo tanto para nosotros la denominación de RAM significará otra cosa, que es la característica fundamental: **Volátil** (no siempre es así, como lo veremos en las memorias **FRAM**).

Las memorias **RAM** son volátiles, es decir, no retienen los datos cuando el suministro de alimentación es interrumpido. Su uso está reservado exclusivamente para almacenar datos temporales con los que efectuaremos cálculos o procesos, pero que no nos interesa guardar indefinidamente.

Debido a su tecnología de almacenamiento, el tiempo de grabación y lectura es muy inferior al de sus contrapartes de solo lectura.

MEMORIAS RAM ESTÁTICAS

La forma más antigua y simple de memorias **RAM** son las denominadas estáticas o **SRAM**. Su princi-

pio de funcionamiento radica en el hecho de que están realizadas a partir de una celda básica de memoria: el **flip-flop**.

Es extremadamente veloz, pero dado que un flip-flop suele construirse con **3** a **6** transistores, el espacio ocupado en el chip es muy grande. Por lo tanto, la capacidad de almacenamiento total está limitada a la superficie.

No requiere de circuitos electrónicos externos para el mantenimiento y gestión, como ocurre con las **DRAM** que veremos a continuación.



FIGURA 14. La desventaja es que necesitan de un circuito exterior para realizar el refresco.

En la actualidad, disponemos de memorias **RAM** con interfaz serie, además de las clásicas en paralelo. El tamaño o capacidad de almacenamiento está dado por la cantidad de líneas o pines (bits en el caso de memorias serie) de dirección disponibles por la cantidad de líneas (o bits en seriales) de datos: **2m palabras, o 2m x n bits.** Donde **m** es la cantidad de líneas de dirección y **n** la cantidad de líneas de datos. Por ejemplo, tenemos una memoria con **4 líneas** de dirección y **8 líneas** de datos: **24 palabras de 8 bits (1 byte) = 16 bytes o 24 x 8 = 128bits**.

MEMORIAS RAM DINÁMICAS

Las memorias RAM Dinámicas o **DRAM** utilizan una tecnología de almacenamiento diferente de las **RAM** estáticas, que se basa en celdas capacitivas. Desde este punto de vista la tecnología es similar a la empleada en las memorias **EEPROM**. Sin embargo, dado que la velocidad de escritura de una memoria **RAM** debe ser muy rápida, estos capacitores deben ser lo suficientemente pequeños como para cargarse instantáneamente. A diferencia de la **EEPROM**, el valor de capacidad es tan pequeño que este llega a mantener la carga solo por algunos milisegundos. Por este motivo las memorias **DRAM** necesitan de un refresco permanente para mantener la carga en el capacitor (**Figura 14**).



VENTAJAS DRAM

Esta tecnología permite en un mismo espacio físico lograr matrices de memoria con una capacidad total muy elevadas por sobre una RAM convencional o estática. Otra de las ventajas es que el consumo de potencia es muy inferior.



INFOGRAFÍA 2: GALERÍA VISUAL DE TIPOS DE MEMORIAS

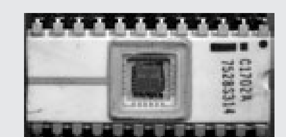
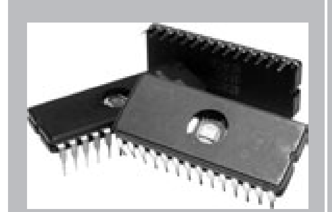
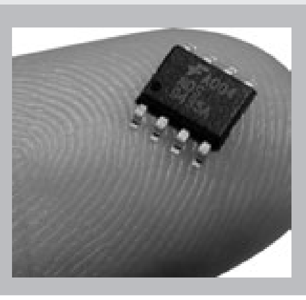
IMAGEN	DETALLES
	**Memoria EPROM** En el año 1971, Intel lanzó la primera memoria EPROM 1702. Era de 2048 bits y podía utilizarse una numerosa cantidad de veces, con la posibilidad de borrarla por medio de rayos UV. Signifcó todo un avance para la época. Imaginemos que si utilizáramos memorias PROM, para cada cambio que hiciéramos en los datos, deberíamos descartar la memoria y reemplazarla por otra programada con la nueva información.
	**Memorias EPROM de la serie 27C** Fueron ampliamente utilizadas por muchos circuitos y aplicaciones electrónicas. Se las puede hallar en controles industriales, control del automóvil e, incluso, en antiguos secuenciadores para realizar efectos luminosos en locales bailables. Es importante destacar que el tiempo de exposición a los rayos UV es de 10 a 15 minutos para lograr un borrado seguro. No debe superarse esa cantidad porque se producirá un envejecimiento prematuro que impedirá futuros borrados.
	**Memoria EEPROM** Éste es un microcontrolador que incorpora una memoria EEPROM y una Flash en su interior. La Flash es utilizada para almacenar el programa que ejecuta el microcontrolador. La EEPROM, en cambio, es para almacenar sólo datos. Por ejemplo, podría guardar valores de temperatura, contraseñas o cualquier dato que pudiera ser de utilidad para el programa que estuviera corriendo en el microcontrolador. Por supuesto, es posible borrar y actualizar los datos que alberga.
	**Eficiencia** Tiene una velocidad comparable a la de las RAM convencionales, pero no es volátil, es decir que sus datos no se pierden cuando se interrumpe la alimentación. Están disponibles para el acceso serie, con lo cual reducen la cantidad de pines necesarios para la conexión. Para lograr el mismo efecto, aún hoy se utilizan memorias RAM estáticas con un pequeña pila o batería de respaldo, de modo que, en caso de que se interrumpa la alimentación, siga energizada, como sucede con las BIOS de las PCs.

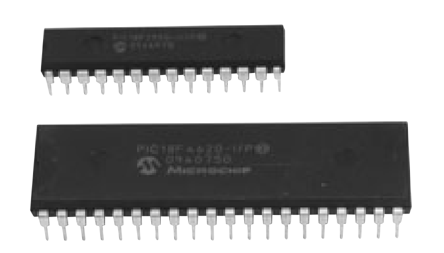


IMAGEN	DETALLES
	**Memoria SDRAM** Es una memoria RAM dinámica, pero, a diferencia de las DRAM, que son asincrónicas (el cambio de los estados depende exclusivamente de la memoria), en éstas (sincrónicas) el cambio de estado depende de una señal externa que le marca el ritmo, denominada clock o reloj. Es la CPU la que decide cuándo grabar o leer, y cuánto esperar.
 	**Memoria RDRAM** La denominación proviene de Rambus DRAM. Es una memoria rediseñada por completo y que no hereda ninguna tecnología de sus contemporáneas SDRAM. A pesar de su ancho de palabras de 16 bits, trabaja a una velocidad mayor que las SDRAM. Es utilizada en muchos videojuegos. También está disponible en módulos de 32 y 64 bits.
	**Memoria Flash** El pequeño espacio ocupado, y la alta velocidad de lectura y escritura de las memorias Flash las hace especialmente indicadas para utilizar en pen drives, tarjetas de memoria, etc. No se ven alteradas por vibraciones o golpes externos. En la actualidad, existen memorias Flash de 64 GB en un tamaño de 14 x 18 x 1,8 mm.
 	**SSD o Solid State Drive** Es una unidad de disco rígido de estado sólido. Si bien el término "drive" no signi?ca exactamente "disco", su uso es como el de un disco rígido, que reemplaza a los normales electromecánicos. Está construido sobre la base de memorias Flash, y alcanza una capacidad de 64 GB y más. Tiene muchas ventajas en comparación con los discos duros convencionales, como mayor velocidad de lectura y mayor constancia, no dañarse por golpes, ser silencioso y tener menor consumo.

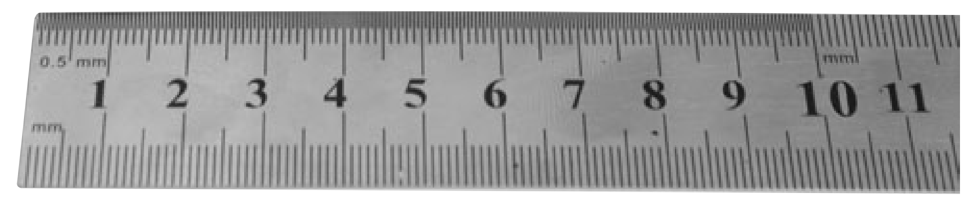
Memorias FRAM

La memoria **FRAM** (**RAM Ferroeléctrica**) es una memoria de estado sólido, que tiene un funcionamiento más parecido a las antiguas memorias de ferrita. Veamos de qué se trata.

Recientemente han aparecido en el mercado memorias **RAM** no volátiles, basadas en la propiedad **ferroeléctrica**. Para ciertas aplicaciones en donde el costo no sea un impedimento, es la panacea de las memorias, puesto que es tan veloz como una memoria RAM tradicional y no pierde los datos al interrumpir su alimentación.







Las memorias FRAM pueden construirse en tamaños pequeños, pero con gran capacidad de almacenamiento

Su principio de funcionamiento está basado en la sensibilidad al campo eléctrico de la estructura cristalina, que polariza a un átomo de esta en uno de dos estados posibles y que se mantiene estable aunque se retire el campo eléctrico. Por este motivo, la memoria FRAM es una memoria no volátil.

Este efecto recuerda a las antiguas memorias **MRAM** empleadas en sistemas de cómputos en los años 50 y 60, donde se aprovechaba la propiedad magnética de los núcleos de ferrita. Sin embargo, el principio es radicalmente distinto, pudiendo las memorias **FRAM** construirse en tamaños pequeños y brindando una gran capacidad de almacenamiento.

Las memorias **FRAM** tienen como desventaja que la lectura es **destructiva**, es necesario aplicar un campo para conocer el estado del átomo móvil y esto lo altera. Pero esto no es un problema, ya que la circuitería de control realiza el procedimiento de acceso dos veces y restablece la situación original.

Estas memorias están disponibles en los modos de acceso paralelo y serie, y en muchos casos son pin a pin compatibles con memorias **RAM** y **EEPROM** comunes del mercado, pudiendo reemplazarlas. La retención de datos es de 10 años. Algunas memorias **FRAM** representativas son: **FM1808**, **FM24C64** (**Figura 15**).

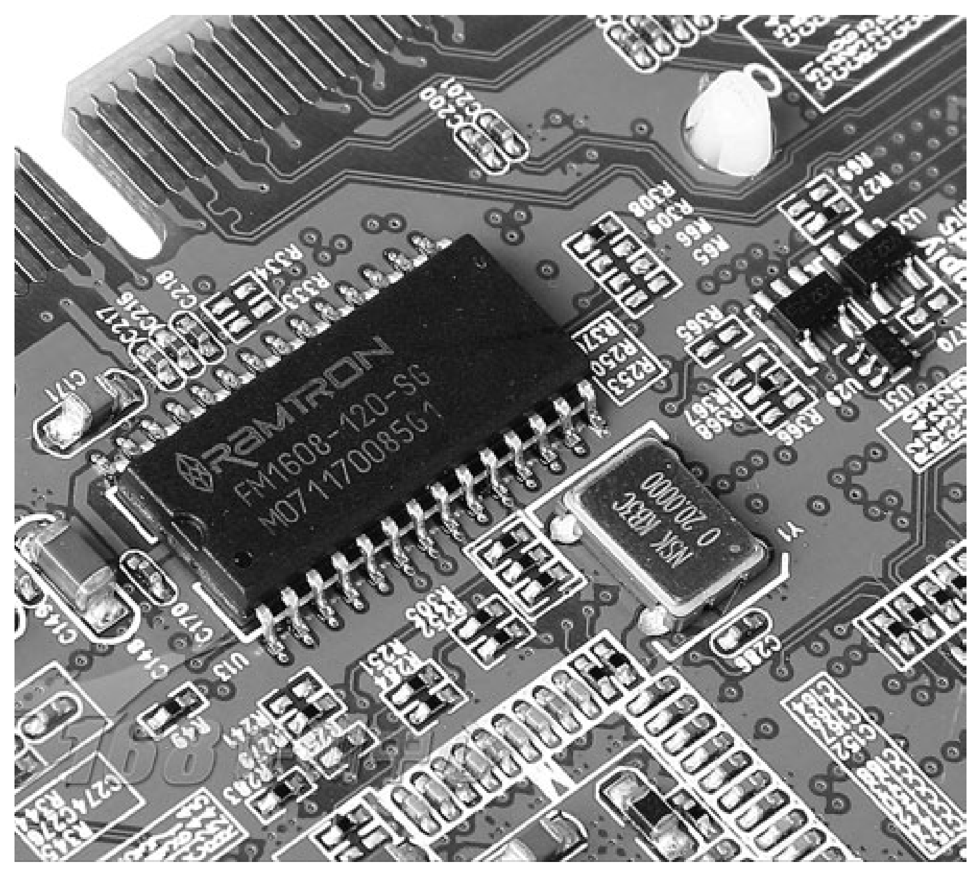


FIGURA 15. Memoria FRAM. En la actualidad el uso y recomendación es emplear memorias serie, dejando solo las paralelas en los casos de reemplazo de partes ya existentes.

Multiple **choice**

1 ¿A qué tipo de almacenamiento pertenece el disco rígido?

a- Magnético.
b- Óptico.
c- Electrónico.
d- Ninguno de los anteriores.

2 ¿A qué tipo de almacenamiento pertenece la memoria ROM?

a- Magnético.
b- Óptico.
c- Electrónico.
d- Ninguno de los anteriores.

3 ¿A qué tipo de almacenamiento pertenece el Blue-ray?

a- Magnético.
b- Óptico.
c- Electrónico.
d- Ninguno de los anteriores.

4 ¿Cómo se llama la unidad mínima de almacenamiento?

a- Fila.
b- Columna.
c- Celda.
d- Tabla.

5 ¿Cuál de los siguientes medios de almacenamiento tiene memoria volátil?

a- DVD.
b- Memoria RAM.
c- Disquete.
d- Disco rígido.

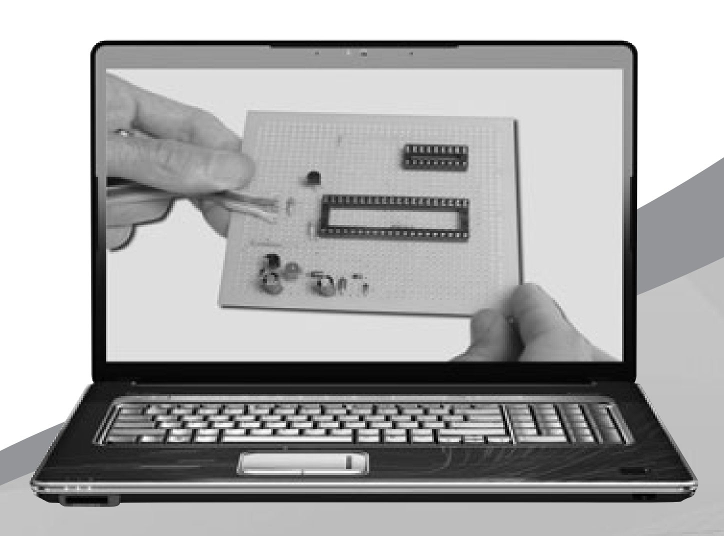
6 ¿Cómo es la memoria EPROM?

a- Volátil.
b- No volátil.
c- Volátil y no volátil.
d- Ninguna de las opciones anteriores.

Respuestas: 1 a, 2 c, 3 b, 4 c, 5 b, 6 b.

Capítulo 4

Microprocesadores y microcontroladores



Estudiaremos las diferencias entre ellos y realizaremos un programador casero para microcontroladores.

¿Cómo funcionan?

Aunque no los veamos, los microprocesadores y microcontroladores están en todas partes. Son unos chips omnipresentes capaces de resolver tareas de diversas complejidades. Sus aplicaciones pueden ser infinitas y están en la industria, la robótica, la domótica y las comunicaciones.

Es por eso que dedicamos este capítulo a comprender sus conceptos y diferencias. Veremos cuáles son las partes que integran un microprocesador y cómo ejecutan las instrucciones para realizar una tarea específica. Luego, nos introduciremos en el mundo de las microcomputadoras, para analizar cuáles son los bloques que las integran y las distintas arquitecturas que podemos hallar.

Haremos una revisión de los distintos lenguajes de programación para los microprocesadores, y aprenderemos el concepto de interrupciones y sus aplicaciones.
Los microprocesadores y los microcontroladores han cambiado la forma de pensar y diseñar los circuitos electrónicos. Desde que Intel lanzó en 1971 el **8080**, el primer microprocesador exitoso, estos dispositivos no han dejado de evolucionar, y hoy en día es imposible imaginarse la vida sin ellos.

Los microprocesadores son circuitos integrados complejos encargados de realizar diferentes tareas

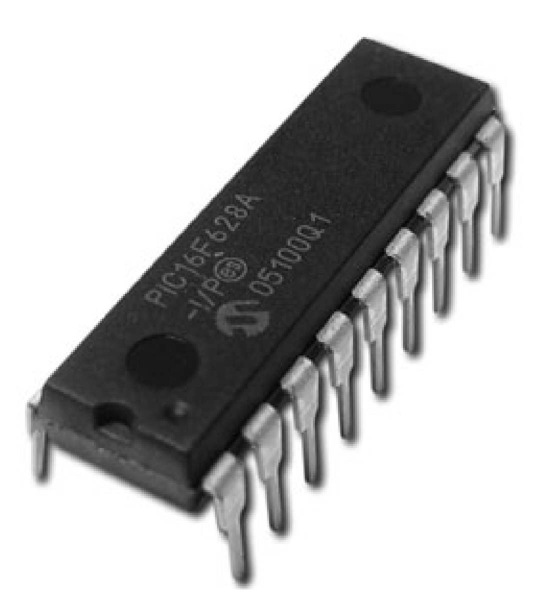


FIGURA 1. **Los** microcontroladores PIC **son uno de los dispositivos más utilizados en el área de control. Aquí podemos ver uno de los encapsulados que ofrece el fabricante.**

Pero a esta altura del tema nos surgen las primeras preguntas: ¿qué son los microprocesadores? ¿Y los microcontroladores? ¿Cómo podemos diferenciar unos de los otros?

¿QUÉ ES UN MICROPROCESADOR?

Los microprocesadores son circuitos integrados que contienen millones de transistores en su interior, los cuales crean circuitos complejos encargados de realizar diferentes tareas. También se los denomina **unidad de procesamiento central o CPU**, ya que muchos de ellos pueden actuar como el "cerebro" de un sistema computacional, administrando todas

las tareas que este realice y llevando a cabo las operaciones con los datos.

Los microprocesadores están diseñados para interpretar y ejecutar las instrucciones que nosotros les indiquemos y que suelen ser operaciones simples, como sumar, restar, multiplicar y dividir. Pero también existen instrucciones lógicas, como **AND**, **OR**, **NOT**, entre otras. El listado de instrucciones recibe el nombre de **programa**, que las ejecuta una por una por medio del microprocesador.

La potencia, el tamaño y la complejidad de los microprocesadores fueron incrementándose con el correr del tiempo. Tanto es así que hoy en día podemos observar procesadores que integran millones de transistores en su interior y con varios núcleos para aumentar su capacidad de procesamiento (**Figura 1**).

DIAGRAMA BÁSICO DE UN MICROPROCESADOR

En la **Figura 2** podemos observar el diagrama básico de un microprocesador. El elemento principal es la **ALU** (unidad aritmética lógica), que se encarga de llevar a cabo todas las operaciones lógicas y aritméticas que requieran los procesos que se ejecuten. También podemos ver los **registros para almacenamiento temporal** de los datos; el **contador de programa**, que contiene la dirección de memoria de la siguiente instrucción por ejecutar; un **registro de instrucciones** que almacena el código de la instrucción en ejecución y el **bloque de control**.

Todo este bloque de circuitos lógicos realiza dos operaciones de manera continua: la búsqueda de una instrucción (**fetch**) y su ejecución. El fetch en la memoria de programa es la operación fundamental del procesador y se efectúa de la siguiente manera:

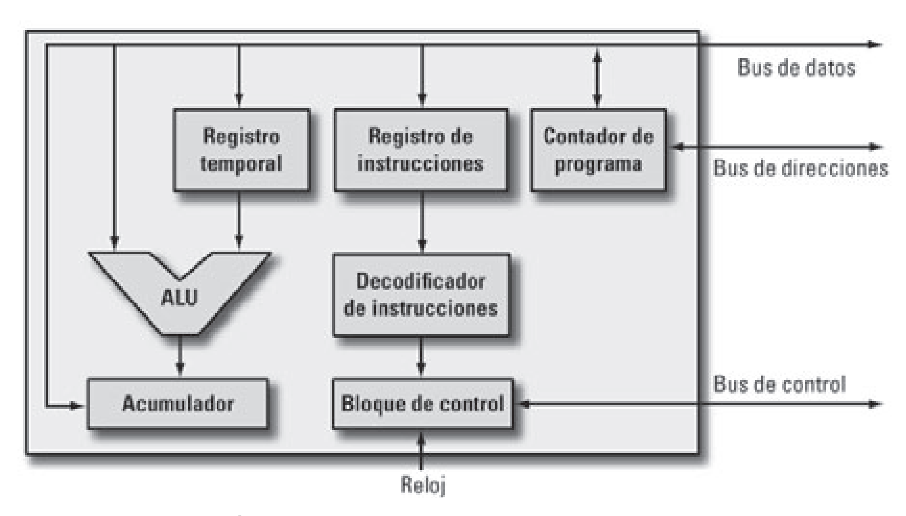


FIGURA 2. Diagrama básico de los componentes que integran un microprocesador.

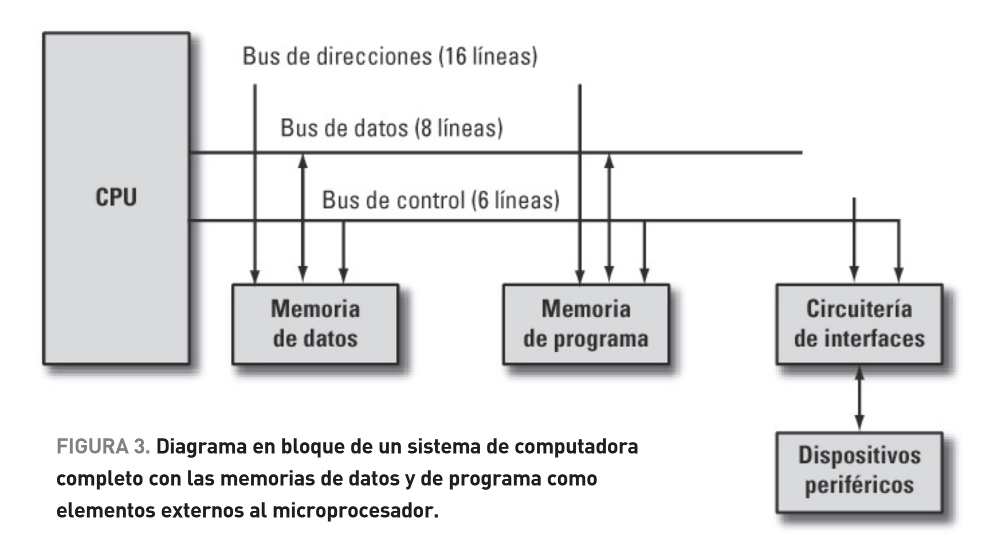


FIGURA 3. Diagrama en bloque de un sistema de computadora completo con las memorias de datos y de programa como elementos externos al microprocesador.

a) El dato que contiene el contador de programa nos indica cuál es la dirección de la próxima instrucción que se ejecutará, y es colocado en el **bus de direcciones**.
b) La unida de control envía una señal de lectura hacia la memoria de programa por el **bus de control**.
c) Los datos contenidos en la dirección de memoria de programa son cargados en el **bus de datos**.
d) Estos datos son procesados para que el código de operación se almacene en el registro de instruc ciones, y los datos, en los **registros de almacenamiento temporal**.
e) Finalmente, el contador de programa se incrementa para volver a buscar la siguiente instrucción.

En la operación de ejecución se decodifica el código de operación, y la unidad de control genera las señales que permiten la entrada y la salida de los registros internos hacia la unidad aritmético-lógica. La **ALU** ejecutará la operación que le indique la unidad de control, y puede devolver el resultado en un **registro interno**, normalmente llamado **acumulador,** o en una **posición** de la memoria de datos (**Figura 3**).

UNIDAD ARITMÉTICO-LÓGICA

La unidad aritmético-lógica se ocupa de realizar las operaciones necesarias con los datos. Ellas son la **suma** y la **sustracción** de números enteros, las operaciones lógicas, como **AND**, **OR** y **NOT,** y las operaciones de **desplazamiento** de bits.

La salida o el resultado de la operación se almacena, como ya dijimos, en un registro interno del microprocesador llamado acumulador. Este registro también puede utilizarse como un operando, con la ventaja de que ganamos en velocidad y eficiencia en las operaciones complejas.

La unidad posee, además, una **entrada de control**, desde donde la unidad de control le indica cuál es la operación por realizar. (**Figura 4**).

Cabe aclarar que en procesadores más complejos podemos encontrarnos con unidades aritmético-lógicas que pueden calcular la multiplicación y hasta la división de los operandos.

CONTADOR DE PROGRAMA

El **contador de programa** (PC) es, básicamente, un **registro contador** que incrementa su cuenta con cada ejecución de una instrucción. El contenido de este registro contador apunta a la dirección donde reside la instrucción que se desea ejecutar en la memoria de programa (**Figuras 4** y **5**).

El valor inicial del contador siempre es la dirección **donde** está la **primera instrucción** del programa.

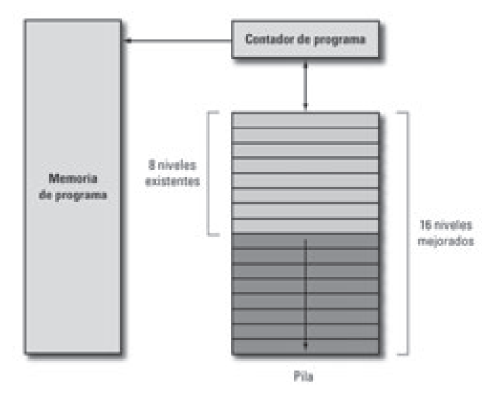


FIGURA 5. El contador de programa siempre contiene la dirección de la próxima instrucción que se ejecutará. Después de un reset del sistema, se inicializa con el valor cero.

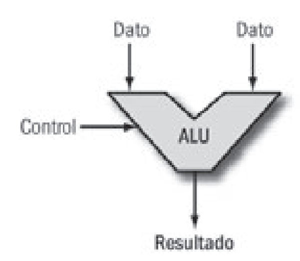


FIGURA 4. Esquema clásico de una unidad aritmético-lógica con las entradas de datos y de control, y una única salida con el resultado.

Además, es posible modificar su contenido para crear saltos hacia rutinas que estén en lugares específicos de la memoria de programa.

La cantidad de **bits** que posee el contador sirve para calcular cuál es la cantidad máxima de instrucciones que puede direccionar el procesador. Con esto tenemos una idea del tamaño de memoria de programa que podemos utilizar. Si tenemos un contador de programa de **16 bits**, podemos direccionar, como máximo, hasta una memoria de **64 KB** (**2^16**).

MEMORIA DE DATOS Y DE PROGRAMAS

Los programas y los datos que controlan la ejecución de las instrucciones en un microprocesador necesitan ser almacenados en memorias de datos y de programa.

El contador de programa (PC) es, básicamente, un registro contador que incrementa su cuenta con cada ejecución

MEMORIA DE PROGRAMA

Para realizar una tarea específica, un microprocesador necesita de un programa que le indique, instrucción por instrucción, cuáles son los pasos que debe cumplir. Este programa reside en una memoria externa al procesador llamada **memoria de programa**. Su característica principal es que no debe perder su contenido cuando el sistema carece de energía. Normalmente, se utilizan memorias **ROM** de solo lectura para grabar un programa, porque poseen esta característica tan preciada, aunque tienen la desventaja de que solo pueden ser escritas una sola vez.

Gracias al avance de la tecnología, hoy podemos utilizar memorias **EEPROM** y **Flash** para almacenar código, con la ventaja adicional de que es posible borrar su contenido eléctricamente.

MEMORIA DE DATOS

La **memoria de datos** es también una memoria externa al microprocesador, pero que se encarga de almacenar la información que precisa el procesador para ejecutar las operaciones que le indiquemos. El tipo de memoria que se emplea para los datos es la **RAM**, porque puede almacenar datos temporales que pueden ser escritos y leídos una infinidad de veces. Además, son de acceso rápido, por lo que el sistema gana en velocidad y eficiencia.

También es posible querer almacenar datos que no se pierdan luego de que el sistema se quede sin alimentación. En este caso, se pueden utilizar memorias no volátiles, como las **EEPROM** de baja capacidad, ideales para guardar contraseñas o nombres de usuarios.

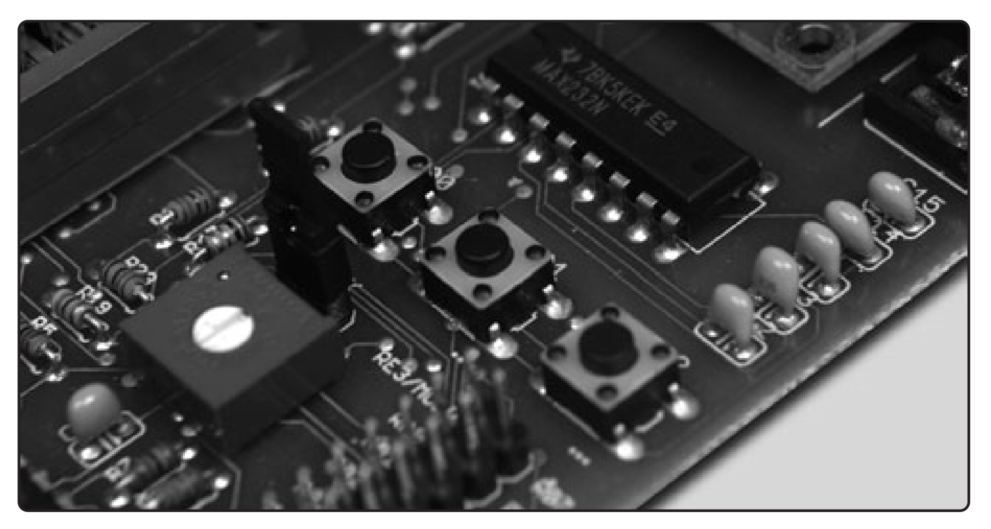


FIGURA 6. El pulsador es el ejemplo más simple de un periférico de entrada. Si armamos una matriz de pulsadores, podemos crear un sencillo teclado para ingresar datos.

Unidades de **entrada y salida**

Para que un procesador pueda comunicarse con el mundo externo necesita de unidades de entrada y de salida que codifiquen los mensajes para interpretarlos. A las unidades que funcionan como interfaz entre el mundo externo y el procesador se las llama **periféricos**. Los periféricos se comunican con el procesador mediante los **buses** de **dirección**, de **datos** y las señales de **control**. Existe dos formas de transmitir información entre un periférico externo y el procesador: en **paralelo** y en **serie**.

Le transmisión en **paralelo** utiliza todas las líneas de comunicación del bus de datos, y no requiere realizar ningún sincronismo entre el periférico y el procesador. Por su parte, la que es en **serie** hace la transformación de paralelo a serie y transmite el byte, bit por bit. Este tipo de trasmisión necesita de un sincronismo entre el **procesador** y el **periférico**. Los periféricos pueden clasificarse de forma general en: periféricos de **entrada** y de **salida**

PERIFÉRICOS DE ENTRADA

Se ocupan de codificar los mensajes o señales del exterior para que el procesador pueda interpretarlos. El ejemplo más sencillo de un periférico de entrada es el **teclado**, con el cual un usuario puede introducir un programa o datos. Pero esto no solo se limita al ingreso de datos por parte de un usuario, sino que también es posible recibir datos de una aplicación de control. Los dispositivos de monitoreo, como los **sensores,** son periféricos de entrada, ya que pueden convertir distintas magnitudes, como el calor o la presión, en señales que una computadora sea capaz de leer (**Figura 6**).

PERIFÉRICOS DE SALIDA

Permiten observar los resultados arrojados por el procesador de una manera más cómoda que si se presentaran como unos y ceros. La pantalla y la impresora son los periféricos de salida más conocidos, pero también tenemos dispositivos actuadores que afectan de manera mecánica todo lo que los rodea, como los motores y los relés (**Figura 7**).



FIGURA 7. **Las pantallas LCD son muy utilizadas como periféricos de salida para mostrar los resultados de las operaciones del procesador.**

PROGRAMACIÓN DE MICROPROCESADORES

La programación de microprocesadores puede realizarse entre tres tipos de niveles básicos de lenguajes: en **código máquina**, **ensamblador** y de **alto nivel**.

LENGUAJE EN CÓDIGO MÁQUINA

Es el lenguaje elemental del microprocesador, pero el más complicado de utilizar. Cada instrucción posee **códigos hexadecimales** que son específicos de ese procesador. Esto hace que la programación de las distintas familias de microprocesadores sea incompatible. Solo se trabaja en código máquina con algunos periféricos que disponen de un repertorio determinado de comandos.

Todos los lenguajes superiores al final serán transformados a lenguaje máquina para ser introducidos en la memoria, ya que este es el único lenguaje que entienden los microprocesadores. Pero esta conversión no la realiza el programador, sino que existe un software específico para este fin.

Si se desea programar en **código máquina,** hay que entender previamente a fondo el microprocesador que se va a utilizar, ya que cada bit de cada instrucción tiene un significado concreto y es muy fácil equivocarse.

El lenguaje ensamblador utiliza nemotécnicos para las mismas instrucciones en binario

EL LENGUAJE ENSAMBLADOR

También llamado **assembler**, es un tipo de lenguaje intermedio entre los de **alto nivel** y el lenguaje **máquina**. Cada microprocesador tiene su propio lenguaje assembler, que está en relación directa con su estructura. Este lenguaje usa las mismas instrucciones que posee el microprocesador, solo que el programador no emplea su correspondencia en hexadecimal, como en el lenguaje máquina, sino que utiliza los **nemotécnicos** de dichas instrucciones (**Figura 8**).

SECUENCIA DE EJECUCIÓN DE UN PROGRAMA

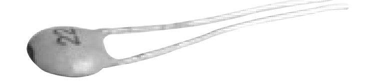
Para el manejo del programa, el microprocesador dispone de dos registros de importancia: el **contador de programa** (**PC**) y el **registro de instrucciones** (**IR**). Para entender el proceso de interpretación del programa, veamos los pasos que se realizan para decodificar una instrucción.



PARA TENER EN CUENTA

Debido a que cada microprocesador tiene su assembler específico, los distintos lenguajes ensambladores de las diferentes familias de procesadores que existen comercialmente no son compatibles entre sí, pese a que emplean instrucciones con idéntico cometido.

Al comienzo de cada instrucción, lo primero que debe hacerse es la lectura del **primer byte** de la instrucción; para esto, el **PC** tiene que direccionar a la posición de memoria en la que se encuentra. Mediante señales de transferencia en el **bus de control**, el byte entra en el microprocesador por el **bus de datos** al registro de instrucciones. Allí es interpretado por el decodificador de instrucciones, mientras el **PC** es incrementado en una cuenta. Si la instrucción es de más bytes, una vez interpretada por el decodificador, se abrirá de nuevo el bus de direcciones y de datos para tomar el segundo byte en la posición de memoria indicada por el **PC** ya incrementado.



El proceso se repetirá otra vez si la instrucción es de tres bytes. Así, según la instrucción sea de uno, dos o tres bytes, la velocidad o tiempo de ejecución será menor o mayor, respectivamente.

Una vez que los bytes de la instrucción han entrado en el microprocesador, este procede a ejecutarla; tras esta operación, vuelve a tomar el byte cuya dirección esté apuntado por el **PC**. Este proceso se repite hasta llegar a la última instrucción (**Figura 9**).

FIGURA 8.

En muchos procesadores se necesita hardware adicional para hacer la programación. Algunos, como el de la imagen, pueden programar los microcontroladores sin sacarlos de la placa.

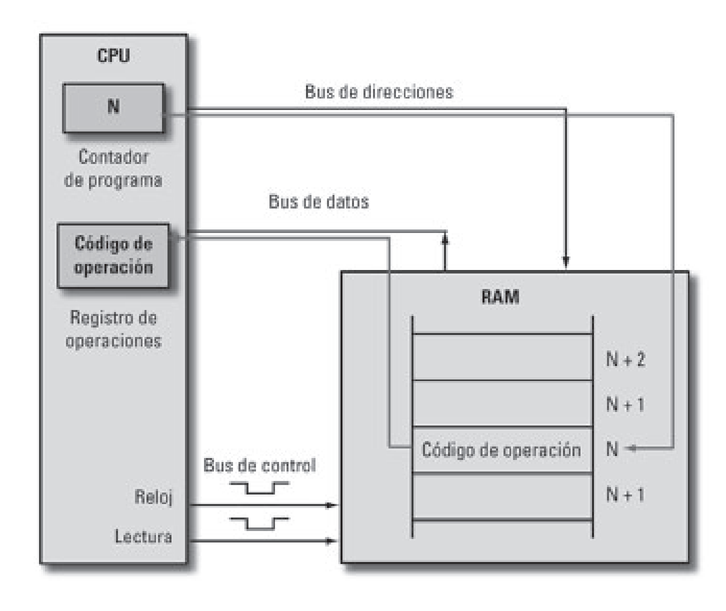


FIGURA 9. Secuencia que lleva a cabo el procesador para buscar una instrucción en la memoria. El PC apunta a la dirección donde se encuentra la instrucción, y la memoria coloca el código de operación en el bus de datos.

LENGUAJES DE ALTO NIVEL

Se llaman de alto nivel porque su sistema de programación está a la altura misma del lenguaje conceptual, matemático y de organización del propio hombre. El desarrollo de los lenguajes de alto nivel fue necesario como consecuencia de la adaptación de la máquina al hombre. Esto trajo muchas ventajas que hicieron que este tipo de lenguaje de programación se impusiera rápidamente. Por un lado, al ser un lenguaje próximo al del hombre –que, en definitiva, es

Los microprocesadores realizan cuatro operaciones básicas para ejecutar una instrucción



EL CICLO DE MÁQUINA

Los microprocesadores realizan una serie de operaciones básicas: búsqueda de la instrucción, decodificación instrucción, ejecución y almacenamiento de los resultados. Estas cuatro operaciones conforman el ciclo de máquina.

quien tiene que programarlos–, permite la reducción de los costos de software, así como también del tiempo de desarrollo. Otras ventajas son su facilidad de aprendizaje, la posibilidad de realizar programación estructurada y el hecho de que para usarlo no es imprescindible tener conocimiento del hardware. Entre los lenguajes de alto nivel, los más conocidos son:

- **MATLAB,** *MATrix LABoratory* (**laboratorio de matrices**): es un lenguaje diseñado para hacer cálculos matemáticos y empleado en el mundo científico y técnico.
- **COBOL,** *Commom Business Oriented Language* (**lenguaje orientado hacia aplicaciones comerciales y de gestión**): es un lenguaje para uso específico en gestión por tener poca capacidad de cálculo, pero con potencia en el manejo de datos.
- **BASIC**, *Begginners All Purpose Symbolic Instruction Code* (**código de instrucción simbólica universal para principiantes**): fue desarrollado por la Universidad de Dartmonth (EE.UU.) para los estudiantes que se inician en el mundo de la programación.
- **C/C++**: Es un lenguaje extremadamente poderoso y eficiente, que nos da la libertad de realizar casi cualquier tarea con una computadora. Es muy popular entre los desarrolladores de software profesional.

Los lenguajes **de alto nivel** también fueron pensados para eliminar la incompatibilidad entre los de **bajo nivel** y los distintos sistemas de procesadores. Sin embargo, esto no es del todo cierto, ya que existen algunas diferencias dentro de un mismo lenguaje de alto nivel con los distintos sistemas que no proporcionan total compatibilidad. Lo cierto es que un programa en lenguaje de alto nivel debe ser traducido a **código máquina**, para lo cual se utilizan programas intérpretes o compiladores (**Figura10**).

FIGURA 10.
En procesadores con alto poder de cálculo, como los DSP (micros con procesamiento digital de señales), resulta muy difícil y costoso programar en assembler.

¿Qué es una **microcomputadora?**

El avance de las escalas de integración permitió integrar cada vez más transistores por unidad de superficie. El desarrollo de la tecnología **VLSI** (**muy alta escala de integración**), que produjo los microprocesadores, pronto permitió introducir en un solo chip todo un sistema mínimo. Esto dio nacimiento a la microcomputadora, actualmente conocida como **microcontrolador**. Dentro de ellas encontramos una **CPU**, una **memoria de programa**, una **memoria de datos**, el **circuito de reset** y el **circuito oscilador**, además de los **puertos de entrada/salida**, también conocidos como **PORTS I/O**.

Las microcomputadoras o microcontroladores nacieron a mediados de los 80 y rápidamente gana-

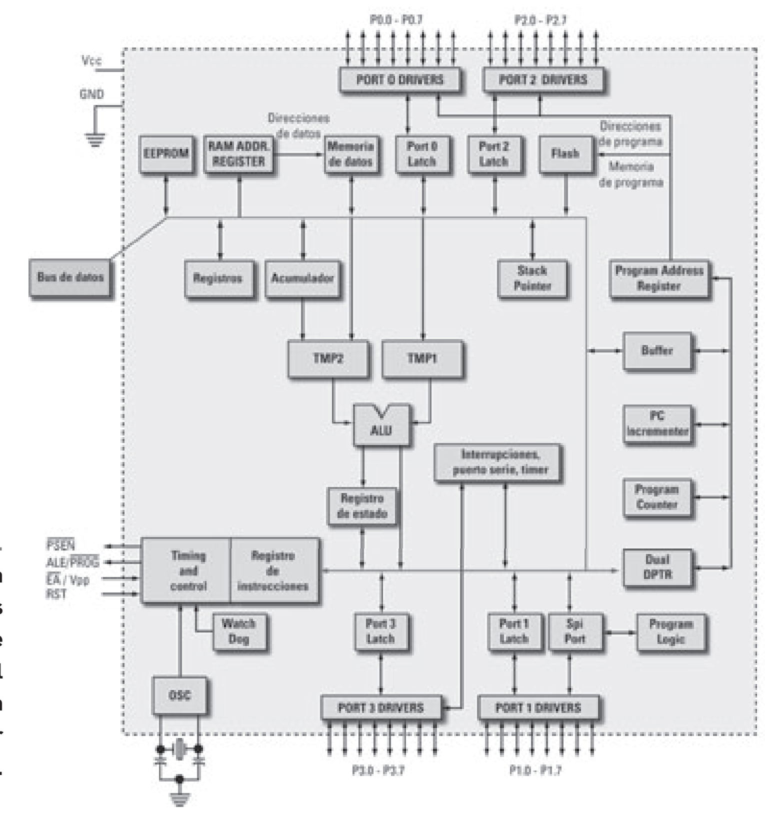


FIGURA 11. **En la figura observamos los bloques que conforman el interior de un microcontrolador básico.**

ron mercado, al desplazar a los sistemas mínimos desarrollados con microprocesadores en el campo del control industrial.

Los **microcontroladores**, debido a su muy bajo costo, alta inmunidad al ruido eléctrico y pequeño tamaño, produjeron la revolución microcontrolada, que desplazó a toda la **lógica cableada** (utilizada en la electrónica industrial) y a la **lógica programada** (realizada con microprocesadores). Es en este campo donde se los bautizó con el nombre de microcontroladores y se desechó el de microcomputadores.

A partir de los **90** los microcontroladores invadieron la electrónica de consumo, brindando a los electrodomésticos y a todo tipo de sistema electrónico de consumo la capacidad de inteligencia y conectividad. El mundo actual está rodeado de microcontroladores; desde nuestros celulares, sistemas de alarmas y lavarropas, hasta las computadoras de abordo de los automóviles. Sin ellos, nuestro mundo actual no existiría (**Figura 11**).

ARQUITECTURA INTERNA DE LOS MICROCONTROLADORES

Mencionamos anteriormente que los microcontroladores están formados por varios bloques. Dentro de ellos existe, como elemento principal, la **CPU,** que se interconecta con sus periféricos para formar lo que se conoce como arquitectura interna. Esta puede ser de dos tipos: **Von Neumann** o **Harvard** (**Figura 12**).

Con el Editor de librerías, es posible crear y editar librerías de componentes, símbolos y encapsulados

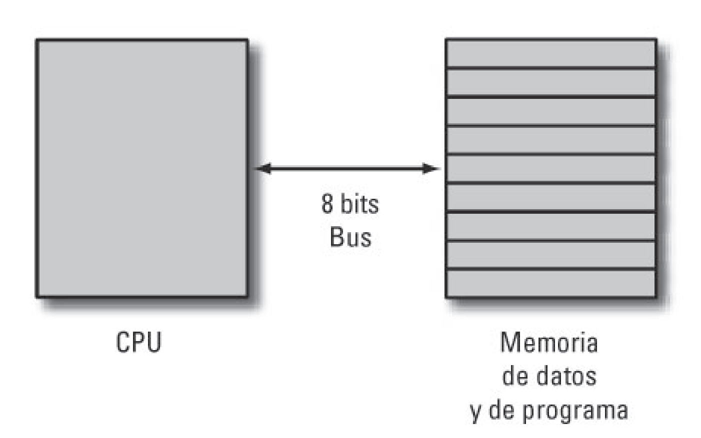


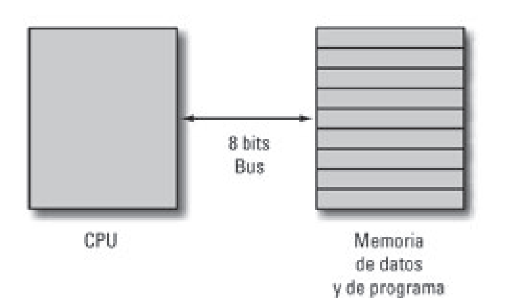
FIGURA 12. En el diagrama podemos ver la conexión entre la CPU, la memoria de datos y la de programa, utilizando el bus de datos para las instrucciones y los datos. Esto genera un cuello de botella.

ARQUITECTURA VON NEUMANN

Fue desarrollada en **1949** por el profesor **John von Neumann** e implementada para la construcción de las computadoras como la **ENIAC** (nombre de la primera computadora electrónica).

Según esta arquitectura, existe un bus de datos que liga la **CPU** con la **memoria de datos** y de **programa** por el cual viajan datos e instrucciones. Este concepto fue muy útil en las primeras décadas de las computadoras, pero al incrementarse la cantidad de datos por procesar, la velocidad de procesamiento se redujo. Rápidamente, la arquitectura se saturó, ya que el **bus** de datos debía compartirse con los datos y las instrucciones, lo que generaba un cuello de botella. Fue así que se presentaron varios problemas en la arquitectura, como mencionamos a continuación.

Por un lado, el ancho del bus de datos era de **8 bits** y, como por él viajan los datos y las instrucciones, el ancho de los datos limitaba el ancho de las instrucciones. Como consecuencia, las instrucciones con más de **8 bits** debían ser enviadas en varias partes, lo cual hacía que el sistema resultara lento.



La arquitectura Von Neumann fue suplantada por los microcontroladores con arquitectura Harvard

Otro conflicto es que nunca se sabía cuánta memoria de programa se usaba, pues esto depende del ancho en **byte** que tengan las instrucciones, lo cual es muy variable.

Esta arquitectura llegó a su fin a finales de los **80** y fue suplantada por los microcontroladores **Harvard**.

ARQUITECTURA HARVARD

Fue desarrollada en 1970 para solucionar los problemas de velocidad de procesamiento que presentaba la arquitectura **Von Neumann** (**Figura 13**).

Esta arquitectura conectó la **CPU** hacia su memoria mediante dos buses distintos: uno de datos y otro de instrucciones. De este modo, el ancho del bus de instrucciones no está limitado por el de datos, y el procesador puede recibir instrucciones por caminos diferentes, aprovechando el tiempo del ciclo de máquina. El concepto y nombre de la arquitectura deviene de la computadora **MARK1**, construida en la Universidad de Harvard en 1944.

En 1975, una compañía americana denominada **General Instruments** formó una división especial dedicada al desarrollo de memorias y microprocesadores:

GI Microelectronics. Esta tomó el concepto de la arquitectura Harvard y lo materializó en su primer microcontrolador denominado **PIC1650,** e introdujo mejoras en el concepto de la arquitectura. Colocó dentro del chip una pila de instrucciones de **dos niveles.** Esto dio la posibilidad de buscar y ejecutar una instrucción en el mismo ciclo de máquina. Se la denominó **arquitectura Harvard modificada**.

En 1985 **GI** vendió a Microelectronics a un grupo inversor denominado Ventura, que reflotó el proyecto del **PIC1650 y** rebautizó la compañía con el nombre de **Microchip**. Ésta desarrolló rápidamente una serie de microcontroladores basados en la arquitectura del PIC. Con el tiempo, otras firmas adoptaron el modelo de Microchip para sus núcleos.

CONCEPTO DE UNA COMPUTADORA

La computadora es un circuito digital capaz de procesar información binaria. En este circuito encontramos un microprocesador, una memoria del tipo **no volátil** (ROM, EPROM, EEPROM, FLASH), una memoria **RAM**, un circuito **oscilador**, un circuito de **reset** y los **puertos de entrada/salida**.

La memoria no volátil es conocida como memoria de programa, ya que en ella se almacena el que hace funcionar a la computadora. La **RAM** es usada por la computadora para guardar los resultados de procesamientos de datos internos o los que provienen del exterior. Por su parte, los **puertos I/O** (entrada/salida) se usan para intercambiar información y controlar todo el entorno externo a la máquina.

Todos estos elementos se encuentran en forma discreta (es decir, en encapsulados en chips individuales) y montados sobre un **PCB**. A este conjunto se lo denomina **sistema mínimo**.

Las computadoras miden su potencia de procesamiento en función de la cantidad de instrucciones que pueden ejecutar por segundo, medida en una unidad denominada **MIPS**.



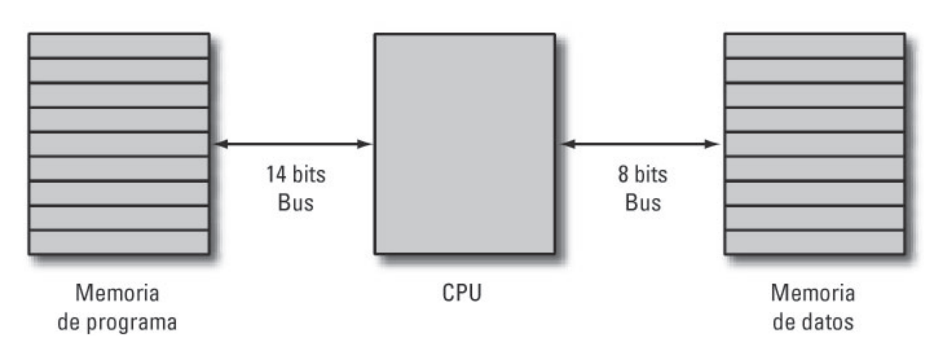


FIGURA 13. **En la figura observamos los buses separados de datos e instrucciones, que pusieron fin al cuello de botella que se generaba en la arquitectura Von Neumann.**

MICROCONTROLADORES Y MICROPROCESADORES

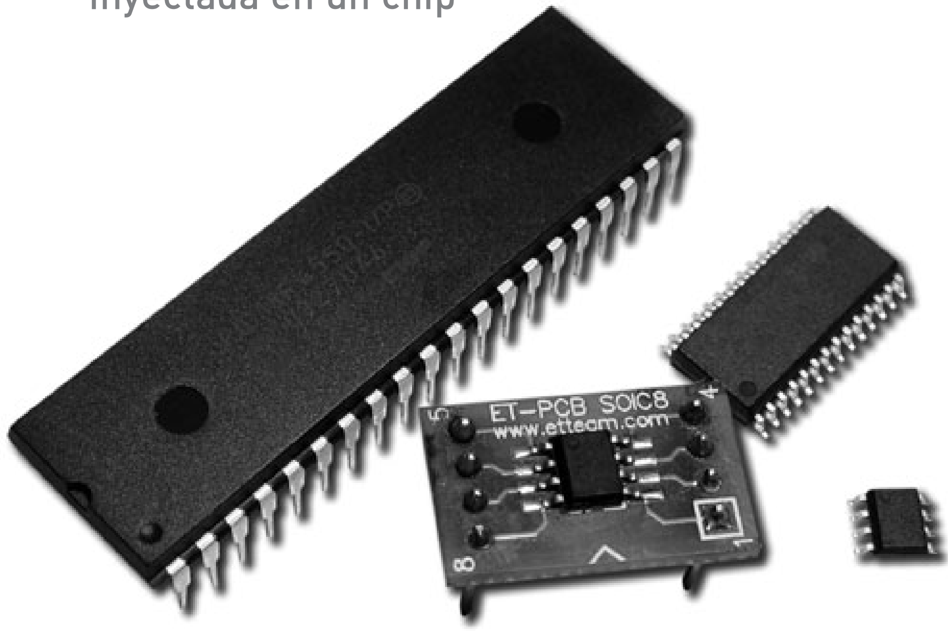
No debemos confundir los microprocesadores con los microcontroladores. Los primeros, simplemente, son la **unidad central de procesamiento**. No incorporan puertos para control de periféricos, ni memoria de programa ni tampoco memoria de datos. Están especialmente diseñados para procesar grandes cantidades de datos y son muy susceptibles al ruido eléctrico.

En cambio, los **microcontroladores** son una pequeña computadora inyectada en un chip. Están diseñados, principalmente, para el control industrial y no para el procesamiento de grandes cantidades de datos. Su principal ventaja radica en la alta inmunidad al ruido, el bajo costo y la reducción de espacio.

Los microcontroladores son una pequeña computadora inyectada en un chip

Las interrupciones

Con las interrupciones podemos hacer que determinados eventos que ocurren en el hardware cambien la rutina del software. Veamos de qué se trata (**Figura 14**).



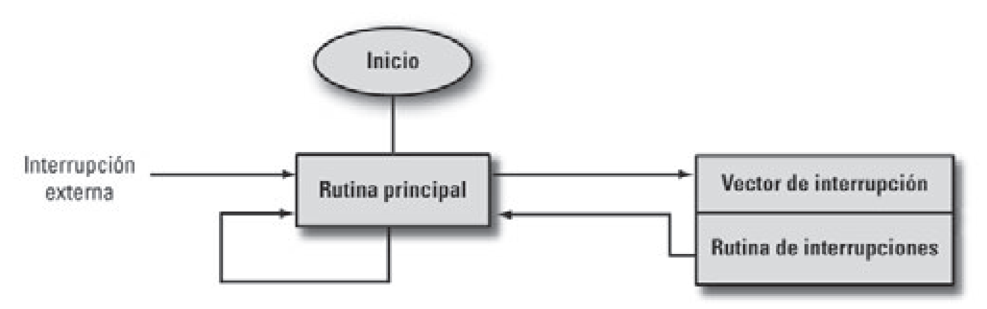


FIGURA 14. En la figura podemos observar el proceso de una interrupción de forma gráfica.

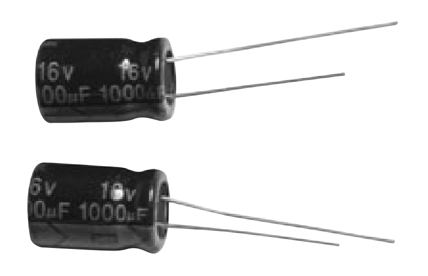
Para entender qué es una interrupción, recurriremos a un ejemplo muy sencillo y que se presenta muy a menudo en nuestros hogares. Cuando enviamos un archivo a la impresora, la rutina principal del programa se encarga de mandar los caracteres al buffer de este equipo. Si la impresora se queda sin papel, el proceso se detiene, y se despliega un mensaje en la pantalla de la PC para advertirnos al respecto.

Como podemos observar, el programa principal se encarga de enviar los caracteres del archivo a la impresora. Sin embargo, si el sensor de falta de papel se activa, se interrumpe la impresión. Este evento hace que el procesador pase a procesar un programa que atienda la interrupción, el cual despliega el mensaje en pantalla.

Una vez que recargamos el papel, el sensor se desactiva, la señal de interrupción desaparece, y el procesador vuelve a ejecutar la rutina principal.

La interrupción es un sistema que provoca un salto en una subrutina, pero disparada por un evento del hardware. Se diferencia de los saltos a subrutina generados por el software, como el producido por la instrucción CALL, en lo siguiente:

- La interrupción es atemporal; el microcontrolador nunca sabe cuándo va a ocurrir.
- Cuando la interrupción ocurre, el procesador abandona el programa que está ejecutando y pasa a procesar una rutina que se encuentra a partir de una posición de memoria fija, denominada vector de interrupción.
- En la interrupción se salva el contenido interno de los registros más importantes del procesador, de modo que luego, al volver al programa que se estaba ejecutando, se prosiga sin alteraciones.



TIPOS DE INTERRUPCIONES

Las interrupciones pueden dividirse en internas o externas, y en enmascarables o no enmascarables. Las internas son disparadas por el hardware interno del microcontrolador, por ejemplo, el conversor analógico/digital, los temporizadores, etcétera. Las externas son disparadas externamente mediante la aplicación de un pulso o un estado sobre un pin del microcontrolador, denominado INT (interrupción).

Pero más allá de si la interrupción es interna o externa, esta puede ser del tipo enmascarable o no enmascarable. Las primeras necesitan tener activado un bit de habilitación para generarse, aunque hayan sido solicitadas; en tanto que las segundas ocurren, sí o sí, no bien sucede el evento de la interrupción.

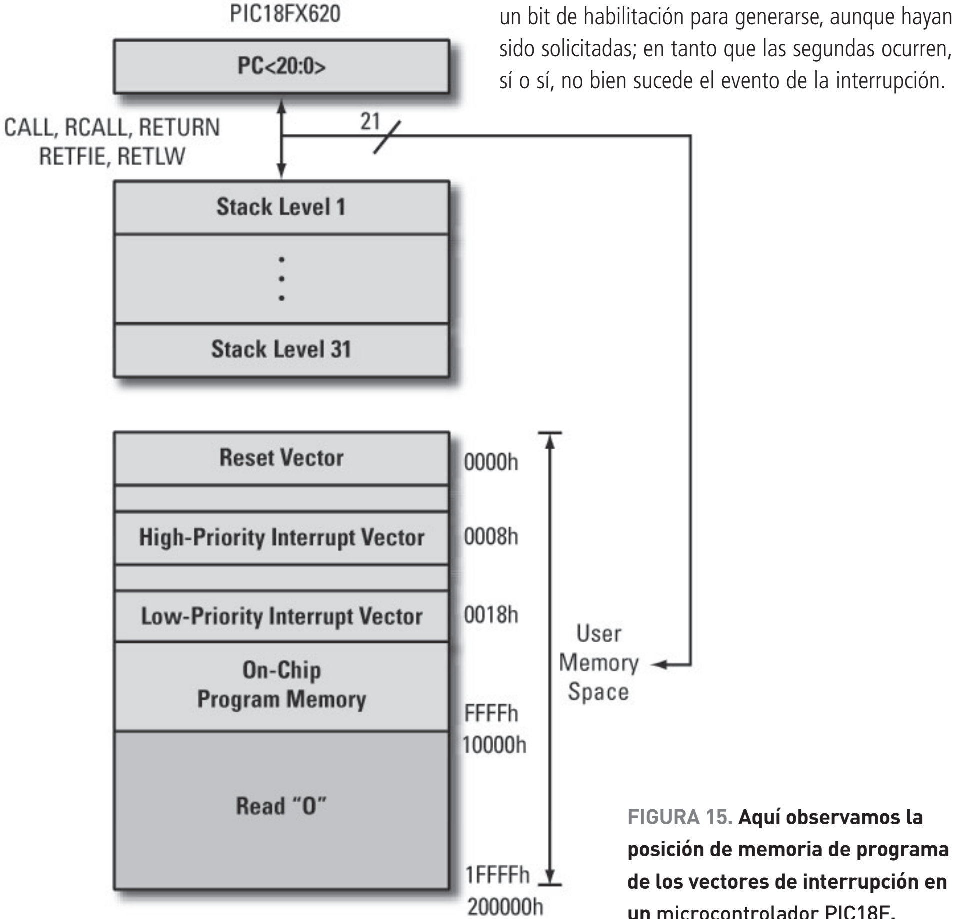


FIGURA 15. Aquí observamos la posición de memoria de programa de los vectores de interrupción en un microcontrolador PIC18F.

La interrupción es un sistema que provoca un salto en una subrutina, pero disparada por un evento del hardware

EL VECTOR DE INTERRUPCIONES

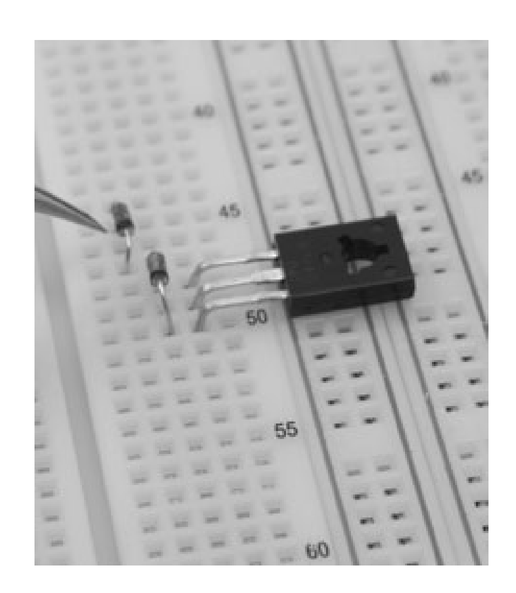
Como hemos visto, cuando la interrupción se genera, el procesador pasa a procesar la rutina que se encuentra a partir de una posición de memoria fija, conocida como vector de interrupciones. En los microcontroladores puede existir más de un vector de este tipo, uno para la interrupción enmascarable y otro para la no enmascarable.

En otros microcontroladores puede ocurrir que cada dispositivo del hardware que interrumpe tenga su propio vector, o que haya vectores distintos según la prioridad que hayamos elegido para la interrupción. Todo depende del microcontrolador que manejemos (**Figura 15**).

Programador para **microcontroladores**

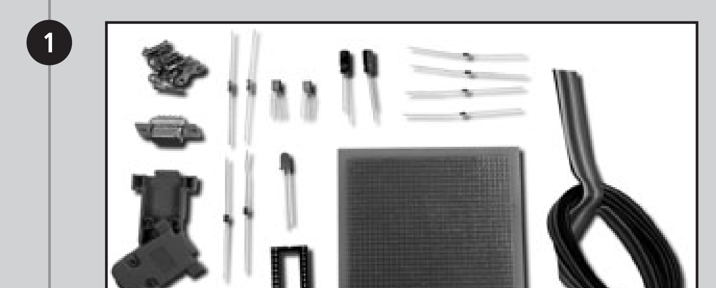
Veremos cómo construir un programador para cualquier PIC que tenga entre 18 y 40 terminales, y que sea de la familia **PIC16F** o PIC18F. Este se conecta al puerto serie de la PC (**RS232**) y se autoalimenta por esa vía, por lo cual no necesita una fuente externa. La ventaja de este dispositivo es que, aunque coloquemos el PIC al revés, no lo quemaremos, porque el puerto no tiene la energía suficiente. Solo precisamos tener en la PC un puerto serie, que es de muy bajo costo.

Para terminar, presentamos el circuito esquemático del programador con todas sus conexiones y componentes (**Figura 16**). En él, los diodos D7 y D5 funcionan como un doblador de tensión, cuyo voltaje es limitado por D6; y C2 almacena la energía que luego provocará la programación del PIC. El voltaje de programación se genera por medio de D4 y D3, y se almacena en C1. El diodo D1 limita la polarización del transistor Q2, y D2 deriva el voltaje negativo para producir un punto de masa virtual.



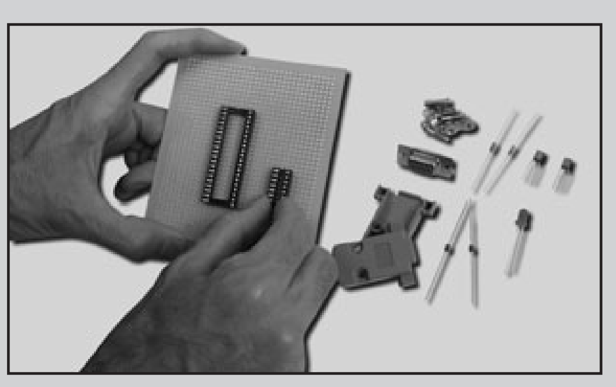
PASO A PASO /1
Armado de un programador para microcontroladores PIC

1



Necesitará un zócalo de 40 pines, otro de 18, resistores de 1K5 y 10K, un transistor BC547 y otro BC557, cuatro diodos 1N4148, un diodo zener de 5V1 y otro 6V2, un LED rojo, dos capacitores electrolíticos de 100 UF x 16 V, un conector DB9 hembra con sus tapas, cable plano de 14 conductores y una placa prototipo de 10 x 10 cm.

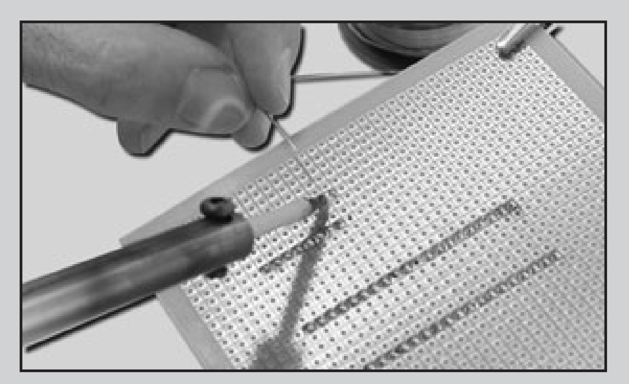




Lo primero que montará serán los zócalos de 40 y 18 terminales, sobre el área que se observa en la imagen. Es importante que los posicione en la misma dirección, para no confundirse y poner los micros al revés.

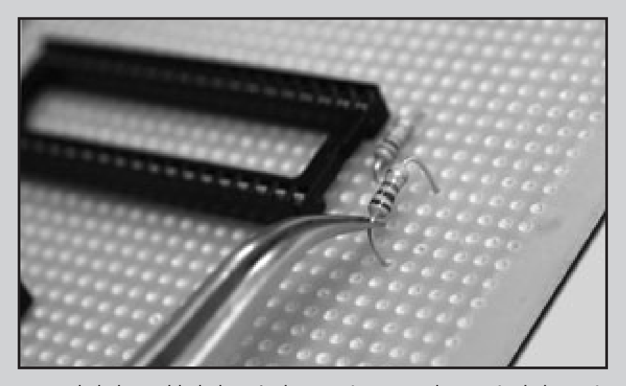
PASO A PASO /1 (cont.)





Acto seguido, suelde los zócalos para evitar que se caigan. En este proceso conviene soldar primero los terminales extremos opuestos, de modo de fijar el zócalo y evitar que caiga al invertir el PCB.

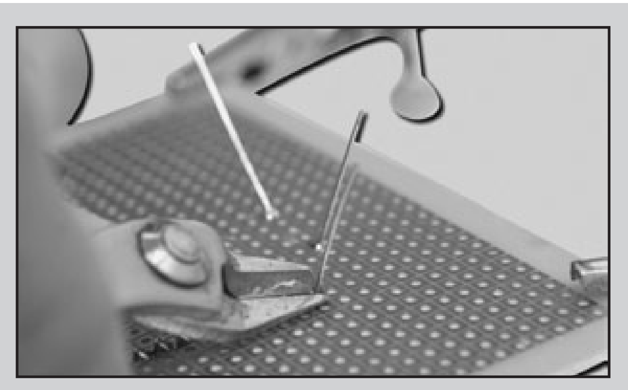




Luego de haber soldado los zócalos, continue con el montaje de los resistores de 1K5 y 10K. Recuerde que el resistor de 1K5 tiene los colores marrón-verde-rojo-dorado, mientras que el de 10K tiene marrón-negro-naranja-dorado.

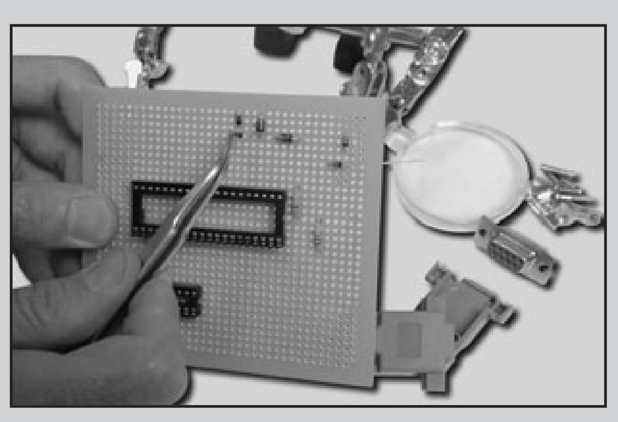
PASO A PASO /1 (cont.)





Para evitar que los resistores caigan al voltear el PCB, al realizar las soldaduras, debe abrir sus terminales a 45 grados y, luego, soldarlas y cortarlas. Es importante que todos los componentes queden con sus cuerpos firmes contra la superficie del PCB.





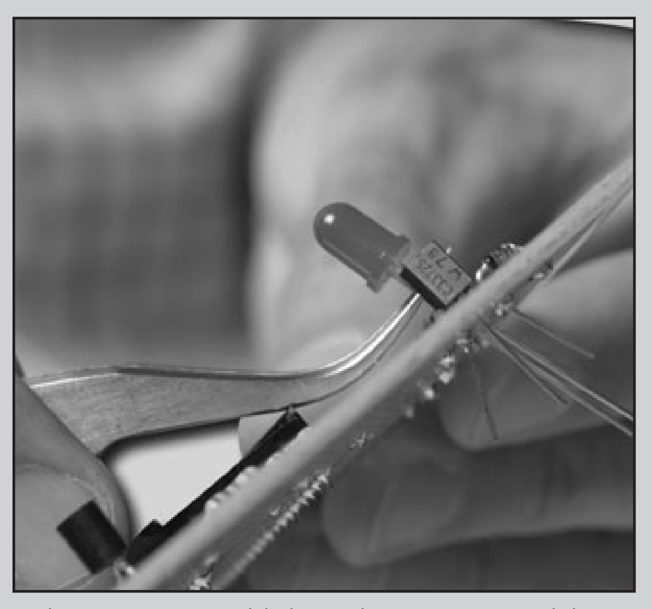
A continuación, monte los diodos 1N4148 y zeners 5V1 y 6V2 aplicando la misma técnica que con los resistores. Preste especial atención a la posición de los cátodos, indicados por la línea negra sobre el encapsulado. De no hacerlo, correrá el riesgo de ponerlos al revés.

PASO A PASO /1 (cont.)

7 Suelde los terminales de los diodos en los semiconductores. Trate de no aplicar el soldador durante mucho tiempo, pues estos elementos son muy susceptibles a la temperatura y pueden deteriorarse.



8



En el siguiente paso monte el diodo LED y los transistores siguiendo las técnicas anteriormente vistas. Hay que tener cuidado con la posición en que los coloque. En el caso del diodo LED, el cátodo se identifica observando el encapsulado, donde hay un polo achatado.



9 Una vez montados el LED y los transistores, proceda a soldarlos. El LED será usado para generar el voltaje de programación e indicar su existencia, mientras que los transistores actúan como switches electrónicos, permitiendo aplicar el voltaje de programación al PIC.

PASO A PASO /1 (cont.)





El siguiente paso es el armado del cable plano sobre la ficha DB9. Para montarla, usará solo seis conductores, que soldaremos a cada terminal de la ficha. Debe hacer coincidir el color, según el código de colores de los resistores, con el número del terminal.





Una vez soldados los conductores, atornille la brida o seguro metálico que evita el deslizamiento y monte el conjunto en las tapas, que le dan cuerpo al conector y protegen las conexiones.

PASO A PASO /1 (cont.)





Monte en el conector los tornillos de fijación con sus respectivos refuerzos metálicos. Estos permitirán fijar luego el conector a la ficha DB9 de la PC. El refuerzo metálico le dará mayor resistencia mecánica a la ficha, pues al atornillarla, los tornillos harán fuerza sobre los refuerzos y no sobre las tapas plásticas.

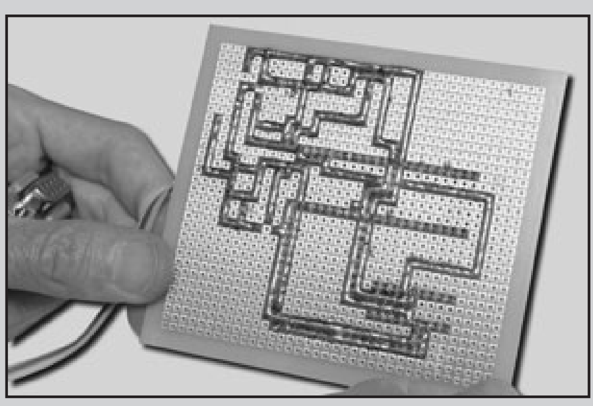




Finalmente, cierre las tapas y coloque los tornillos que las sujetan. Debe ponerlos en contraposición, es decir, haciendo que ingrese uno desde arriba y el otro desde abajo; esto facilita el armado del conector.

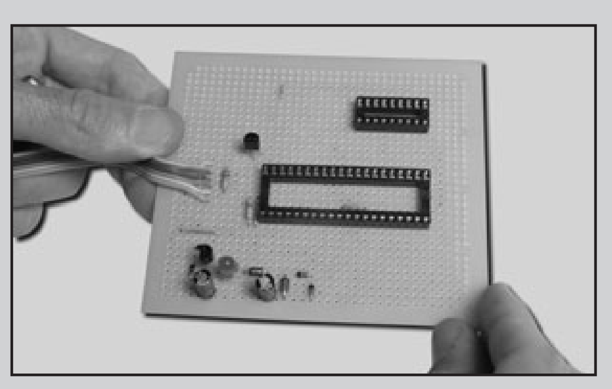
PASO A PASO /1 (cont.)





Una vez efectuado el montaje de la ficha, suelde las conexiones entre los distintos componentes, realizando los caminos de estaño. Al completar todas las conexiones, el circuito debería quedar como se muestra en la imagen.





Este es el programador terminado, con todos sus componentes. Se puede ver los zócalos donde luego instalará los microcontroladores por programar; y el LED, que indicará visualmente la fase de programación. El resto de los elementos controlan la programación y generan el voltaje correspondiente.

FIGURA 16. El transistor Q2 actúa como llave programadora, y Q1, como llave para generar los pulsos de clock.

Multiple **choice**

1 ¿Qué nos permite que un microprocesador pueda comunicarse con el mundo externo?
a- Las memorias de datos y programas.
b- Las unidades de entrada y salida.
c- El ensamblador.
d- El código máquina.

2 ¿Dónde deben almacenarse los programas y los datos que controlan la ejecución de las instrucciones en un microprocesador?
a- Las memorias de datos y programas.
b- Las unidades de entrada y salida.
c- El ensamblador.
d- El código máquina.

3 ¿Cuál de las siguientes no es una memoria de programa externa?
a- RAM.
b- ROM.
c- Flash.
d- EEPROM.

4 ¿Cuál de las siguientes es una memoria de datos?
a- RAM.
b- ROM.
c- Flash.
d- EEPROM.

5 ¿Cuál de los siguientes no es un periférico de entrada?
a- Teclado.
b- Sensores.
c- Pantalla.
d- Mouse.

6 ¿Con cuál de los siguientes tipos de niveles básicos de lenguaje no se puede programar un microprocesador?
a- Código máquina.
b- Ensamblador.
c- Alto nivel.
d- Bajo nivel.

Respuestas: 1 b, 2 a, 3 a, 4 a, 5 c, 6 d.

Capítulo 5

Microcontrolador PIC16F



Estudiaremos el circuito de alimentación y el de reset. Además veremos los puertos de entrada y de salida.

Microcontroladores **PIC16F**

Los **PIC16F** son unas de las familias de microcontroladores de **8** bits más populares en el mercado. Son fabricados por la firma **Microchip** y se ofrecen con una amplia gama de funcionalidades, destacándose por su sencillez y bajo consumo.

Los microcontroladores PIC16F pertenecen a la familia de rango medio de los micros de 8 bits de Microchip. Poseen un repertorio de 35 instrucciones, procesador de filosofía RISC (Set de instrucciones reducidas), buses de arquitectura **Harvard**, construidos en pastillas de tecnología **CMOS**, frecuencia de operación hasta **20MHz**, hasta **8KB** de memoria Flash de programa, conversores **A/D** integrados y manejo de interrupciones.

Estos microcontroladores PIC son ideales para aprender las bases de la programación de estos dispositivos.

MICROCONTROLADOR PIC16F887

Este microcontrolador pertenece a la familia **PIC16F88X**, la nueva familia de microcontroladores de rango medio de Microchip. Posee un encapsulado de 40 pines del tipo DIP (Dual In-Line Package) y la disposición de sus pines la podemos ver en la **Figura 1**.

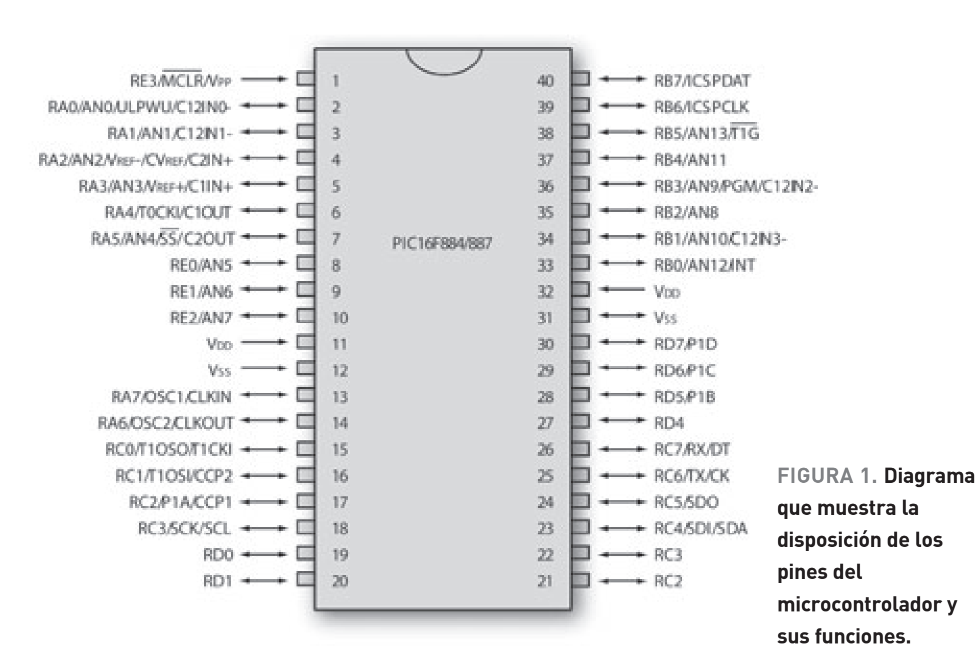


FIGURA 1. **Diagrama que muestra la disposición de los pines del microcontrolador y sus funciones.**

El PIC16F887 posee las siguientes características:

CPU:
- Tecnología RISC de 35 instrucciones.
- Todas las instrucciones se ejecutan en un ciclo de instrucción, salvo las instrucciones de salto que se ejecutan en dos.
- Frecuencia de operación de hasta 20MHz.

Memoria:
- 8 K x 14 bits de memoria flash de programa.
- 368 bytes de memoria RAM de datos.
- 256 bytes de memoria EEPROM para datos.
- Lectura/escritura de la CPU a la memoria flash.
- Stack de hardware de 8 niveles.

Otros:
- Modo SLEEP de bajo consumo de energía.
- Programación y depuración serie "In-Circuit" (ICSP).
- Rango de voltaje de operación de 2 a 5,5 volts.
- Rangos de temperatura: comercial, industrial y extendido.
- Conversor A/D de 10 bits de resolución.
- 2 Timers de 8 bits y 1 de 16 bits.
- 35 Pines de entrada/salida digital.
- Oscilador interno de gran precisión.

CIRCUITO DE ALIMENTACIÓN

El microcontrolador PIC16F887 se alimenta con 5 V, que deben provenir de una fuente que mantenga una alimentación estable y confiable. Los pines de alimentación para VDD se encuentran en la posición 11 y 32 y para VSS los pines 12 y 31. En la **Figura 2** se puede ver un esquema de la conexión de la alimentación del PIC.

Es importante colocar un capacitor de desacople de **0,1 µF** entre los pines de **VDD** y **VSS** para evitar que los transitorios generados por un componente no ingresen a la fuente, afectando a otros dispositivos. Cuando alimentamos a 5 V y el **PIC** funciona con un cristal de **4 Mhz**, el consumo es inferior a los **1,5 mA**. Si reducimos la frecuencia de trabajo a **32 KHz** el consumo típico es menor a **40 µA**.



FIGURA 2. Circuito de alimentación para el microcontrolador PIC16F887. Los 5 V pueden provenir de una batería o de una fuente regulada.

OSCILADOR

El PIC16F887 permite hasta 8 diferentes modos para el oscilador. El usuario puede seleccionar alguno de estos 8 modos modificando un registro especial de configuración. Este registro se localiza en la dirección 2007H de la memoria de programa y se lo utiliza para configurar los recursos de PIC.

Los modos de operación para el oscilador son:

- RC: Resistencia / Capacitor externos
- LP: Cristal para baja frecuencia
- XP: Cristal para frecuencias intermedias
- HS: Cristal para altas frecuencias.

El microcontrolador PIC16F887 posee dos osciladores internos independientes

Los tres modos LP, XT y HS usan un cristal resonador externo, la diferencia sin embargo es la ganancia de los drivers internos, lo cual se ve reflejado en el rango de frecuencia admitido y la potencia consumida. En la tabla 1 se muestran los rangos de frecuencia así como los capacitores recomendados para un oscilador en base a cristal.

Lo más común es que utilicemos un cristal externo de 4MHz con capacitores de 22pF.

Oscilador Interno

El PIC16F887 posee dos osciladores internos independientes que puede ser controlado por medio de registros internos. Uno de los osciladores trabaja a bajas frecuencias y el otro en altas frecuencias. Este último está calibrado de fábrica en 8 MHz y su frecuencia puede ser alterada por software. Podemos ver más información de este modo de operación en las hojas de datos del microcontrolador.

MODO	FRECUENCIA TÍPICA	CAPACITORES RECOMENDADOS	
		C1	C2
LP	32KHz	68 a 100 pF	68 a 100 pF
	200KHz	15 a 30 pF	15 a 30 pF
XP	100KHz	68 a 150 pF	68 a 150 pF
	2MHz	15 a 30pF	15 a 30pF
	4MHz	15 a 30pF	15 a 30pF
HS	8MHz	15 a 30 pF	15 a 30 pF
	10MHz	15 a 30 pF	15 a 30 pF
	20MHz	15 a 30 pF	15 a 30 pF

TABLA 1. **En la tabla podemos ver los diferentes rangos de frecuencias que soporta el PIC para un cristal externo. Es importante respetar el rango de valores que recomienda el fabricante para los capacitores que acompañan al cristal.**

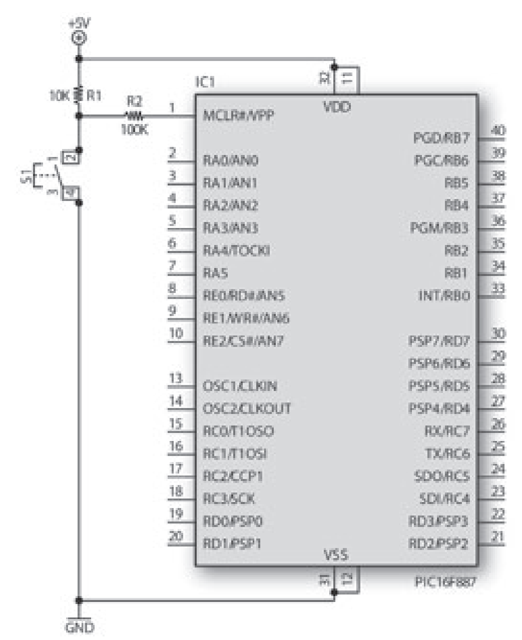


FIGURA 3. La imagen muestra un circuito típico para el reset**. Esta configuración aplica un nivel bajo en el pin** MCLR **generando un reset del sistema.**

Oscilador Externo

También es posible conectar una señal de reloj generada mediante un oscilador externo a la patita OSC1 del PIC. Para ello el PIC deberá estar en uno de los tres modos que admiten cristal (LP, XT o HS).

Circuito de Reset

Cuando se produce un **reset** se reinicializa el procesador, el programa en ejecución se abandona y el contador de programa se carga con la dirección 0. En esa dirección comienza el programa de la aplicación.

Las formas más comunes para hacer un reset son:

- Quitar y volver a establecer la alimentación del microcontrolador. Cuando se conecta la alimentación se produce automáticamente un reset. Este tipo de reset se lo conoce como *Power-On Reset* (**POR**).
- La segunda opción es aplicar un nivel lógico bajo en el terminal **MCLR** (*Master Clear Reset*). Mediante este terminal podemos provocar exter

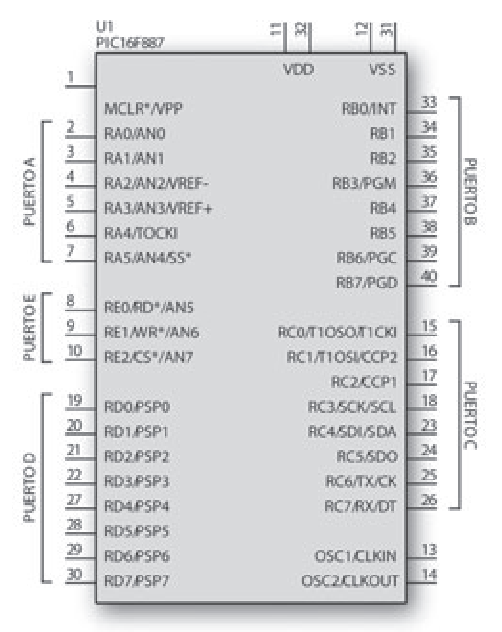


FIGURA 4. Conexión de un cristal externo a los pines OSC1 **y** OSC2 **del microcontrolador y los drivers internos que generan una señal de** CLOCK **que sincroniza el sistema.**

namente un reset en el momento que queramos, sin necesidad de desconectar el microcontrolador de su alimentación. Par introducir un nivel bajo en este pin debemos colocar un pulsador que lleve al pin MCLR a ese estado lógico. En la **Figura 3** podemos ver un esquema típico para el circuito de reset.

PUERTOS DE ENTRADA Y SALIDA

El microcontrolador PIC16F887 posee cinco puertos de entrada y salida, los puertos **A, B, C, D y E**. Debido a que el bus de datos es de 8 bits, es lógico que el tamaño de los puertos sea el mismo. Es así que los puertos B, C y D tienen 8 líneas de E/S digitales, mientras que el puerto A tiene 6 líneas (que por software puede llegar a ser 8) y el puerto E solo tres líneas. Cada pin de cada puerto puede configurarse de manera individual como entrada o salida digital para recibir o enviar datos, según lo necesitemos.

Pero no todos los puertos son solamente E/S digitales, sino que además tienen funciones que se encuentran **multiplexadas** para ahorrar pines en el encapsulado. Por ejemplo, el puerto A puede ser utilizado además como entrada analógica para el conversor A/D incluido en el integrado, mientras que el puerto **B** tiene los pines para programación en **RB6** y **RB7** y una entrada para interrupción externa en el pin **RB0** como funciones más importantes. (**Figura 4**).

Además es importante saber que cada puerto tiene una capacidad limitada para entregar corriente a la salida o para recibirla, y el fabricante la especifica en las hojas de datos del dispositivo.

El microcontrolador PIC16F887 posee cinco puertos de entrada y salida, los puertos A, B, C, D y E, los cuales tienen el mismo tamaño

Dirección	
0000h	Vector de Reset
...	...
0004h	Vector de Interrupción
0005h ... 07FFh	Página 0
0800h ... 0FFFh	Página 1
1000h ... 17FFh	Página 2
1800h ... 1FFFh	Página 3

FIGURA 5. **La memoria de programa posee cuatro páginas de 2 K, completando un total de 8 K para posiciones de memoria. En la imagen podemos ver el vector de reset y el vector de interrupciones.**

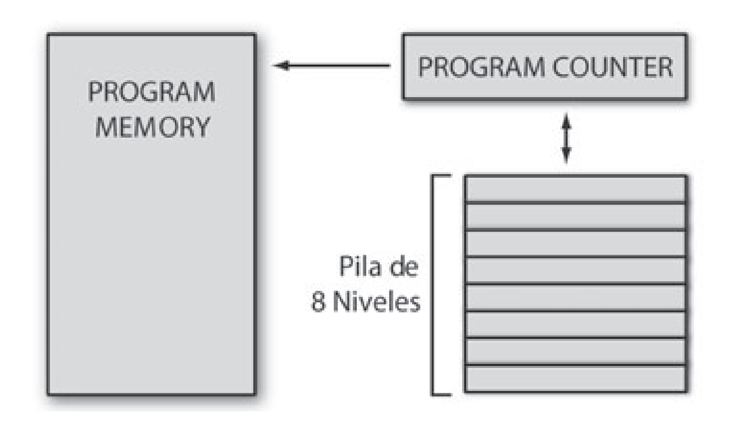


FIGURA 6. En la figura vemos el contador de programa **y la** pila **o** Stack de 8 niveles**. El contador siempre contiene la dirección de la próxima instrucción para ejecutar.**

ORGANIZACIÓN DE LA MEMORIA

El **PIC16F887** tiene dos tipos de memoria: **memoria de datos** y **memoria de programa**, cada bloque de memoria posee su propio bus: el bus de datos y el bus de programa; obteniendo así una arquitectura Hardvard. Esta arquitectura permite que se tenga acceso a cada bloque de memoria durante el mismo ciclo de instrucción. Estas dos memorias tienen funcionalidades diferentes y por ello su tecnología de fabricación es diferente.

La memoria de programa

La memoria de programa almacena las instrucciones de la aplicación. Está realizada con memoria del tipo Flash, lo que permite la escritura y el borrado las veces que sea necesario.

Las instrucciones de los **PIC16F88X** (familia a la que pertenece el PIC16F887) tienen una longitud de 14 bits, por lo que las posiciones de memoria deben tener la misma longitud. El PIC16F887 en particular tiene 8192 posiciones de memoria para programa (8 K) por lo que para acceder a una de las posiciones necesitaríamos un bus de 13 líneas (2^{13}= 8 K). Este es el tamaño de bus que Microchip utiliza para los PIC de rango medio.

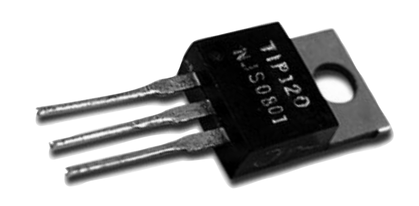
La memoria de programa está dividida en páginas de igual tamaño. En los PIC16F88X las páginas son de 2 K. Así, el PIC con el que trabajaremos tendrá cuatro páginas. (**Figura 5**).

La página 0 es común en todos los PIC de esta familia y tiene solamente dos posiciones que se encuentran reservados:

- **Vector de reset** (ubicado en la posición 0000H): cuando se conecta la alimentación o se resetea el procesador, la primera instrucción que se ejecuta es la que se encuentra en esta posición. Todas nuestras aplicaciones empezaran en el vector de reset.
- **Vector de interrupción** (ubicado en la posición 0004H): cuando se produce una interrupción el contador de programa siempre apunta a esta dirección, entonces el comienzo del programa que atiende a las instrucciones deberá situarse en esta posición.

	File Address		File Address		File Address		File Address
Indirect addr. [1]	00h	Indirect addr. [1]	80h	Indirect addr. [1]	100h	Indirect addr. [1]	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	SRCON	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	CM1CON0	107h	BAUDCTL	187h
PORTD [2]	08h	TRISD [2]	88h	CM2CON0	108h	ANSEL	188h
PORTE	09h	TRISE	89h	CM2CON1	109h	ANSELH	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2 [1]	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUB	95h		115h		195h
CCPR1H	16h	IOCB	96h	General	116h	General	196h
CCP1CON	17h	VRCON	97h	Purpose	117h	Purpose	197h
RCSTA	18h	TXSTA	98h	Reglaters	118h	Reglaters	198h
RCREG	19h	SPBRG	99h	16 Bytes	119h	16 Bytes	199h
TXREG	1Ah	SPBRGH	9Ah		11Ah		19Ah
CCPR2L	1Bh	PWM1CON	9Bh		11Bh		19Bh
CCPR2H	1Ch	ECCPAS	9Ch		11Ch		19Ch
CCP2CON	1Dh	PSTRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General	3Fh	General Purpose		General Purpose		General Purpose	
Purpose	40h	Reglaters		Reglaters		Reglaters	
Reglaters		80 Bytes		80 Bytes		80 Bytes	
96 Bytes	6Fh		EFh		16Fh		1EFh
	70h	accesses	F0h	accesses	170h	accesses	1F0h
	7Fh	70h-7Fh	FFh	70h-7Fh	17Fh	70h-7Fh	1FFh

FIGURA 7. La memoria de datos está dividida en cuatro bancos **de** 128 posiciones **cada una. Al principio de cada banco vemos los registros de funciones especiales y luego los registros de propósitos generales.**



Contador de Programa

Este registro contador incrementa en una unidad su valor cada ciclo de instrucción, excepto cuando hay una instrucción de salto o una interrupción, en estos últimos casos puede cargarse con cualquier otro valor. (**Figura 6**).

En caso de que ocurra esto último, el valor que tiene el contador de programa antes de realizar el salto debe ser almacenado para retomarlo posteriormente. Para ello utiliza una pila **Lifo** (*Last Input First Output*, **Último en entrar Primero en salir**) de 8 niveles, de forma que guarde la última dirección antes del salto en el nivel 1 de la pila, y la pueda retomar en el momento que la necesite.

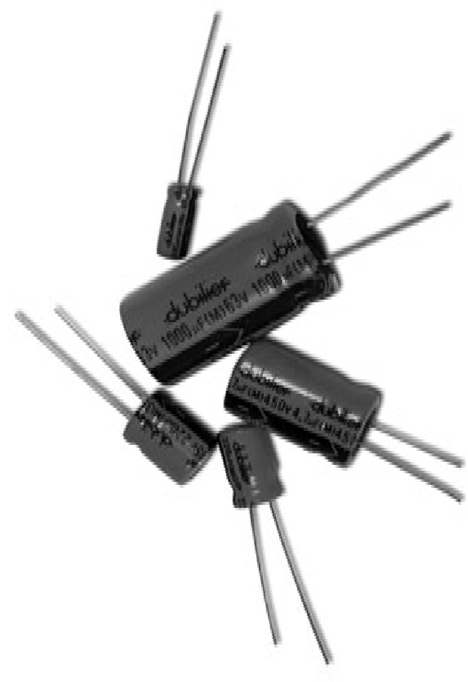
La pila o **Stack** se carga y descarga por el nivel 1, por lo que cuando se carga una dirección en este nivel, el contenido de cada nivel pasa al siguiente. Cuando se descarga el nivel 1 en el contador de programa, el contenido de cada nivel pasa al precedente.

Las interrupciones y la instrucción **Call** guardan el contenido del contador de programa en la pila y las instrucciones **Return**, **Retfie**, y **Retlw** cargan el contenido de la pila al contador de programa.

Memoria de datos

En la memoria de datos se almacenan los datos de propósito general y los registros específicos cuyos bits controlan el funcionamiento del microcontrolador. Como los datos que maneja esta memoria son de carácter temporal, la memoria de datos está formada por memorias RAM. También el PIC tiene la capacidad de almacenar datos que queremos conservar al desconectar la alimentación, estos datos son guardados en una zona especial de memoria **Eeprom** que posee el PIC.

La zona de memoria de datos se compone de cuatro bancos de **128** posiciones de datos de **8** bits cada uno. Cada banco posee Registros de **Propósito General** (**GPR**) que almacenan datos que utiliza el programa y **Registros de Funciones Especiales** (**SFR**) que contienen bits de control para el procesador y de los periféricos del PIC. (**Figura 7**).



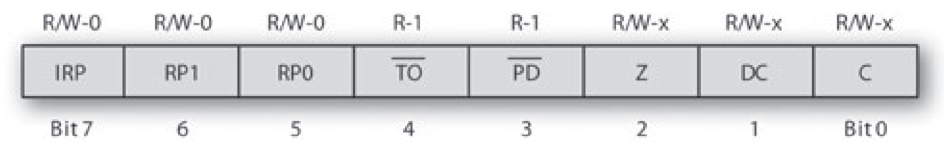


FIGURA 8. Registro de estado, llamado STATUS **y sus bits.**

REGISTROS DE PROPÓSITO GENERAL

Cuando ejecutamos una instrucción, el procesador puede requerir de datos que se encuentran almacenados en la memoria de datos. Si los datos requeridos son utilizados por el usuario, estos deben ser almacenados en la zona de registros de propósito general (abreviados GPR). Entonces la zona de datos GPR almacena los bytes que utiliza el usuario de manera libre para propósitos generales. En la figura podemos visualizar los bloques para los registros GPR; en el banco 0 tenemos 96 bytes para almacenar este tipo de datos que comienza en la posición **20h** hasta **7Fh**, en el banco 1 tenemos 80 bytes desde **A0h** hasta **EFh**, y en los bancos **2** y **3** volvemos a tener **96** bytes para datos. En las últimas posiciones de los bancos **1**, **2** y **3** tenemos un bloque de memoria de **16** posiciones que son registros que están mapeados, o mejor dicho duplicados, a los últimos 16 registros del banco 0. Es decir que si intentamos leer o escribir algún registro en las últimas posiciones de los bancos 1, 2 y 3 , lo que realmente estaremos haciendo es modificar los registros del banco 0. Estas posiciones son ideales para los datos que se utilizan de manera frecuente, porque nos evitamos realizar los cambios de bancos para direccionar hacia esos registros.

Registros de Funciones Especiales

Las primeras posiciones de los bancos del área de memoria de datos son ocupadas por los **Registros de Funciones Especiales** (abreviados **SFR**) y sirven para propósitos específicos en el funcionamiento y la configuración del microcontrolador. Cada registro SFR esta asociado a una función general y dentro de cada uno, cada bit tiene una función específica. Es muy importante aprender la función y el manejo de cada uno de estos registros para poder realizar un uso correcto del microcontrolador.

La gran mayoría de estos registros están dedicados al manejo de los periféricos o recursos que posee el PIC, pero otros están dedicados a almacenar el estado del procesador, el direccionamiento de memoria, el control de las interrupciones y el manejo de los puertos de entrada y salida digital.

El registro de estado

Este registro ocupa la cuarta posición en los cuatro bancos de memoria y es llamado **Status**. Es un registro de control que en su interior almacena el estado de las operaciones aritméticas de la **Alu**, el estado del **Reset** y los bits para la selección del banco de la memoria de datos.

Veamos en detalle las funciones de cada uno de sus bits.

- **Bit C** (**carry**): es el bit de acarreo, sirve para indicar si se produjo un acarreo en el bit más significativo en la última operación.
- **Bit DC** (**Digit Carry**): es el bit de acarreo de digito, es similar al bit C, salvo que indica el acarreo entre los bits 3 y 4. Se lo utiliza cuando realizamos operaciones con datos en BCD.
- **Bit Z** (**Zero**): es el bit de cero, se activa cuando el resultado de una operación aritmética o lógica es cero.
- **Bit /PD** (**Power Down**): bit de apagado, se pone en cero cuando se ejecuta una instrucción Sleep (modo bajo consumo).
- **Bit /TO** (**Time Out**): bit de fin de tiempo, se pone en cero cuando ocurre un desborde del perro guardián (temporizador que resetea el micro cuando termina su cuenta).
- **Bits RP1**:**RP0**: son los bits de selección del banco de registros.
- **Bit IRP**: bit para la selección de bancos, utilizado en el direccionamiento indirecto.

El registro de Contador de programa

El contador de programa del **PIC16F887** posee un registro mapeado en la memoria de datos. Como el contador de Programa contiene 13 bits y las posiciones de memoria son de 8 bits, el contador esta representado por dos registros llamados **PCL y PCLATH**. El registro PCL contiene los 8 bits menos significativos del **PC** y se encuentra en la posición 02h, pero los 5 bits de la parte alta no están mapeados directamente en la memoria de datos, sino que utiliza un registro que sirve como latchs para poder escribir estos 5 bits que faltan. Ese registro es el PCLATH.

El registro PCL siempre es actualizado con los 8 bits menos significativos del Contador de programa, en cambio el registro PCLATH nunca se actualiza con los 5 bits más altos del contador (**Figura 9**).

Contador de un dígito con Display de 7 segmentos

Realizaremos nuestro primer programa, para microcontroladores PIC16F887, el cual nos será muy útil para explicar cómo crear un programa en Assembler y cómo funcionan algunas instrucciones.

Para comenzar vamos a explicar que un programa es una secuencia ordenada de instrucciones, las cuales le indicarán a nuestro microcontrolador cómo realizar determinada tarea. Una de las primeras cosas que debemos entender es que el microcontrolador (MCU) nunca se equivoca, en tal caso si el programa funciona mal es el programador se equivocó, pues los MCU realizan fielmente todo lo que les pedimos. Por otra parte la escritura de cualquier programa se realiza a partir de una secuencia de pasos:

- Plantear primero el problema a resolver teniendo bien claras todas las variables que participarán.
- Luego realizaremos de forma gráfica una secuencia de pasos lógicos que nos permitan llegar a la solución (a esto se lo conoce como diagrama de flujo).
- A continuación traduciremos el diagrama de flujo a un lenguaje de programación.

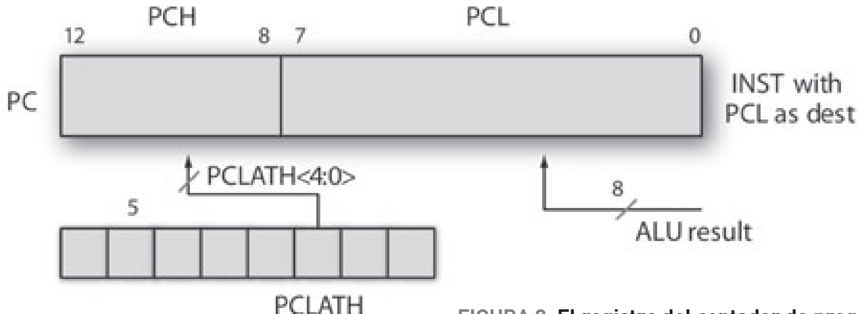


FIGURA 9. El registro del contador de programa ocupa dos bytes**, el registro** PCL **maneja los** 8 bits **menos significativos y el registro** PCLATH **que se utiliza como latch de escritura para los** 5 bits **más significativos del PC.**

- Luego traduciremos el programa escrito a lenguaje de máquina (binario puro) usando un programa denominado **compilador**.

Finalmente cargaremos el programa en lenguaje de máquina en la memoria de programa del PIC usando un **programador**.

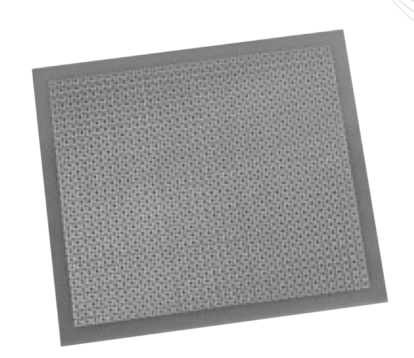
FIGURA 10. En la figura podemos ver el esquema de nuestro contador de 7 segmentos completo.

Los lenguajes **de programación**

Para escribir nuestro programa, podemos usar distintos lenguajes de programación. El lenguaje inmediato es el binario puro, llamado **lenguaje de máquina**, sin embargo las instrucciones en binario son difíciles de recordar y por ello el lenguaje no se usa de forma directa. El lenguaje siguiente es el Assembler el cual reemplaza los códigos binarios por breves sentencias denominadas nemónicos. El Assembler es el lenguaje base pues cada fabricante ofrece compiladores gratuitos para traducir desde este a código de máquina. Pero el Assembler requiere que el programador conozca mucho del hardware del microcontrolador, por ello se desarrollaron los lenguajes de alto nivel. Los lenguajes de alto nivel más usados se denominan lenguaje C y BASIC.

LOS LENGUAJES C Y BASIC PARA PIC

El **lenguaje C** requiere de un conocimiento intermedio del hardware , es muy potente pues permite una gran cantidad de funciones matemáticas y el procesamiento de números decimales, conocidos como números de punto flotante. El **BASIC** necesita un conocimiento elemental del hardware ya que sus instrucciones hacen todo lo que el programador no indica, sin embargo no es tan potente como el C. De esta forma los programadores escriben sus programas en C o en BASIC según su gusto. El compilador de lenguaje C para PIC más usado es el **CCS** para los **PIC16F** y el **C18** para los PIC18F, mientras que en BASIC se puede usar el **PICBASIC**, el **PICBASIC PRO**, el **PICBASIC PROTON** o el **Micro PICBASIC**, cuyos compiladores son fabricados por distintas empresas (**OSHONSOFT**, **MELABS**, etcétera).



CREAR EL HARWARE DEL CONTADOR

El hardware lo haremos sencillo; usaremos un PIC16F887, de muy bajo costo, los pulsos ingresarán por el pin **RA4** provenientes de un sensor externo. Este sensor lo hemos simulado con un pulsador (en la realidad puede ser un microswitch manejado por alguna lengueta o mecanismo). Además podemos ver que el **PORTB** del PIC está conectado a un display de 7 segmentos a LED cátodo común por medio de 8 resistores de polarización de 470 ohm. El esquema lo presentamos en la **Figura 10**.

CREAR EL PROGRAMA DEL CONTADOR

Nuestro programa será sencillo, la parte fundamental del programa consiste en explorar lo que ocurre en el pin RA4, si detectamos un cambio de estado sobre este, entonces incrementamos el valor de un registro llamado **contador**.

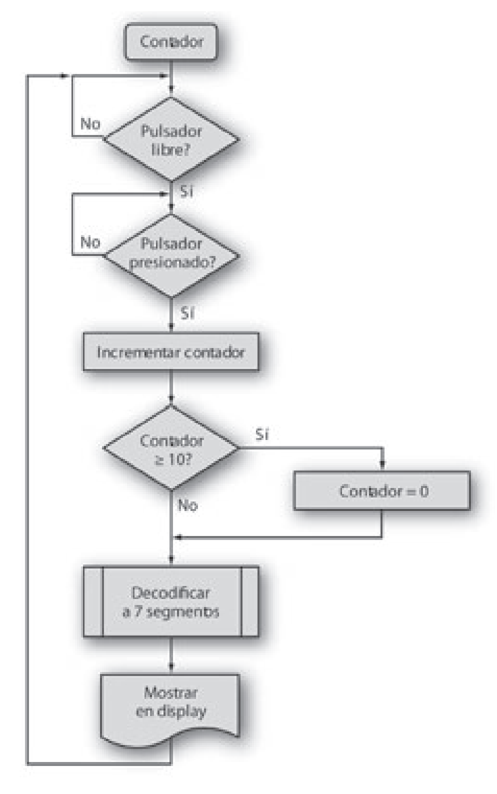


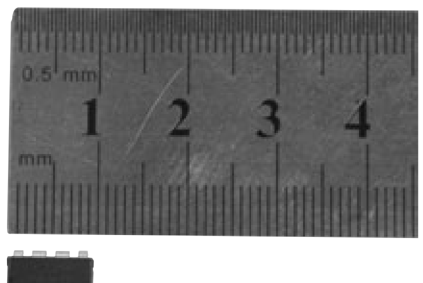
FIGURA 11. Este diagrama de flujo simboliza la operatoria que resuelve el problema. En él pueden apreciarse los distintos símbolos aplicados a cada parte.

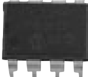
El PORTB del PIC está conectado a un display de siete segmentos

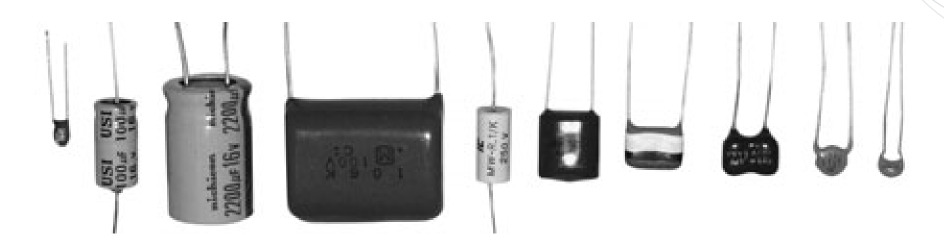
Luego verificamos que nuestro contador no haya sobrepasado la cuenta de 9, si lo hizo lo ponemos en cero, si no lo dejamos tal cual está. Luego, decodificamos el valor del contador y lo transformamos en un valor binario que nos permita dibujar sobre el display el valor del contador. Finalmente volvemos al principio del programa. Hay que observar que lo primero que hacemos en el programa es configurar qué puertos funcionan como entrada y cuáles como salida.

ESCRIBIR EL PROGRAMA EN ASSEMBLER

Para escribir nuestro primer programa lo haremos en Assembler dentro del entorno del MPLAB, lo primero que hay que saber es que el nombre del programa debe tener la extensión **.ASM**, y que se estructura en:







- Una cabecera donde declaramos el modelo del PIC, incluimos la declaración de los registros de funciones especiales incorporando en el código un archivo **.INC** que corresponda con el PIC. Fijamos los fusibles de configuración, luego declaramos los Alias, si se usan, las variables de usuario y el origen del programa principal.
- Las Tablas de constantes donde colocamos tablas de valores constante y mensajes.
- La Rutina Principal donde comienza el programa principal.
- Las subrutinas que son llamadas por la principal.

SET DE INSTRUCCIONES

Es el **conjunto de nemónicos** (instrucciones) del micro, cada uno de los cuales realiza una acción específica. Las instrucciones para operaciones lógicas y aritméticas indican estados especiales mediante flags en el registro de estado. El set de instrucciones puede consultarse en la hoja de datos o en el manual del usuario del micro.

DECODIFICACIÓN

Se realiza mediante una búsqueda en **tabla**. Esto es similar a la operación de una **PROM** visto en el **Capítulo 3**. El número que vamos a convertir nos da la posición (desplazamiento, offset) dentro de la tabla donde está el valor convertido; lo sumamos a la posición de la tabla y obtenemos lo deseado.

A su vez cada segmento se divide horizontalmente en 4 campos a saber:

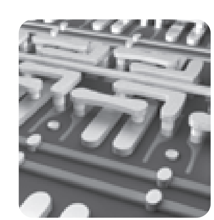
- El campo de las etiquetas, el cual es considera do el primer campo por el compilador. Las etiquetas son puntos de retorno o de llamada que usaremos como referencia en nuestro programa, las cuales siempre deben empezar con una letra y no contener espacios intermedios.
- El campo del Opcode o las Instrucciones donde se colocan los nemónicos de las instrucciones.
- El campo de los operandos donde colocamos los nombres de los registros o constantes que participan en las instrucciones.
- El campo de los comentarios el cual siempre comienza con **";"** y que es usado para escribir un texto orientativo de lo que queremos hacer en esa parte del programa. Esto forma lo que se denomina documentación del programa.





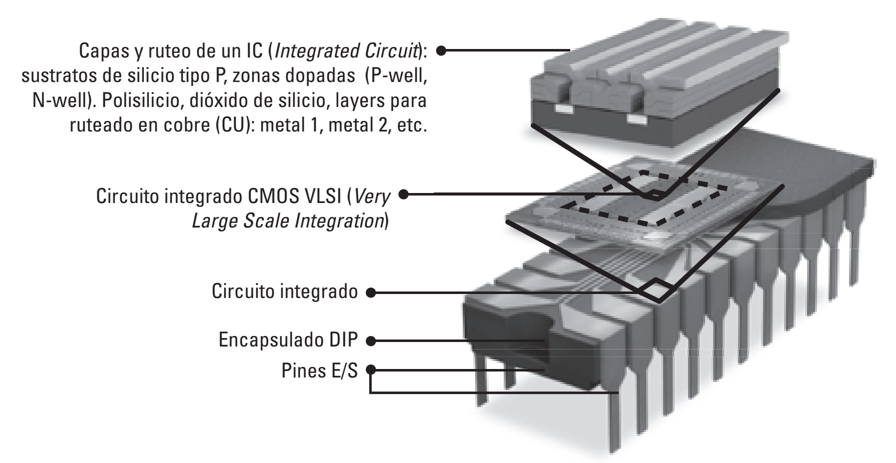
INFOGRAFÍA 3: UN MICROCONTROLADOR PIC16F POR DENTRO

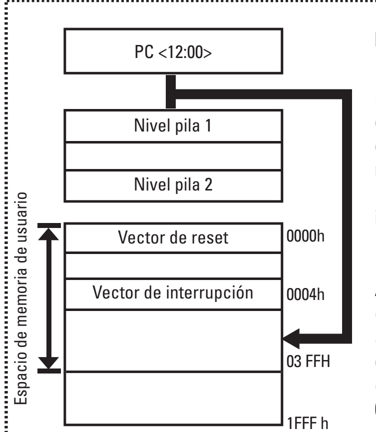
ARQUITECTURA INTERNA DE PIC16F84



Un microcontrolador como el PIC16F84A es un circuito integrado (IC) conformado por un bloque microprocesador o CPU (Unidad Central de Procesos), memoria de programa Flash EEPROM , memoria de datos RAM y puertos de E/S y periféricos.
La CPU o núcleo microprocesador consta de una Unidad de Control (UC), una Unidad Aritmético Lógica (ALU), registros internos e interfaz a memoria. La ALU es la encargada de realizar las operaciones aritméticas básicas (resta, suma, división y multiplicación) y de operaciones lógicas (OR, NOT, AND, etc.).
La Unidad de Control es la que maneja a la ALU enviándole las órdenes en la secuencia en que deben ser ejecutadas y también se ocupa de transportar los resultados obtenidos.

VDD 14
VSS 15
Bus de direcciones de programas
Memoria programa Flash 1kx14
13
Contador de programas 13 bits
8
EEDATA 08h
Memoria de registros RAM 68x8
Bus de programa
Reg. Instrucción
Pila de 8 Niveles 13 bits
Bus de direcciones de datos RAM
EEPROM 68x8
Data EEPROM
8
Dir. MUX
Selec. Banco RAM
EEADR 09 h
7
Direccionamiento directo
8
Direc. Indirec.
01 TMRO
Timer 0
8
Dato lateral
04h FSR
3 RA4 TOCK1
81h Prescaler option TMRO
2 RA3
Datos MUX
1 RA2
OP - Code
Puerto A
18 RA1
MCLR 4
Control y decodificador de instrucciones
ALU
85 h TRISA
17 RA0
Temporizador Power -up
05 h PORTA
13 RB7
Temporizador Oscillator start-up
RPO Z DC C Status 03 h
12 RB6
Q1 Q2 Q3 Q4
8
11 RB5
OSC1CLIKN 16
Power - On Reset
Purto B
10 RB4
OSC2 CLIKN 15
Bloque generación de tiempos
86 h TRISB
09 RB3
Watchdog
08 RB2
W reg
OBH Intcon
06 h PORTB
07 RB1
Sleep
06 RB0 INT
Pines





Memoria FLASH de datos:

Esta arquitectura posee un PC (*Program Counter*) de 13 bits, capaz de direccionar 8 k x 14 bits de memoria de programa. Para este PIC, sólo se implementa físicamente la primera porción de 1 k x 14 bits (0000h-3FFFh). Acceder por encima de esta dirección, causará un acceso circular. El Vector de RESET está en 0000h, y el de interrupciones en 0004h.

	Banco 0	Banco 1	
00 h	Dir. Ind.	Dir. Ind.	80 h
01 h	TMR0	OPTION	81 h
02 h	PCL	PCL	82 h
03 h	STATUS	STATUS	83 h
04 h	FSR	FSR	84 h
05 h	PORT A	TRIS A	85 h
06 h	PORT B	TRIS B	86 h
07 h	-	-	87 h
08 h	EEDATA	EECON1	88 h
09 h	EEADR	EECON2	89 h
0A h	PCLATH	PCLATH	8A h
0B h	INTCON	INTCON	8B h
0C h	68 Registro de propósito general	Mapeados (acceso) en el banco 0	8C h
4F h			CF h
50 h			D0 h
7F h			FF h

Memoria RAM de datos: Está dividida entre el espacio para registros de propósito general (68 GPRs, Dir: 0Ch-4Fh) y los registros de funciones especiales (12 SFRs, Dir: 00h-0Bh). Un bit de control (RP0 bit, del registro: STATUS<5>) permite seleccionar el banco de memoria a utilizar por medio de direccionamiento indirecto. Cada banco posee 128 bytes.

El MPLAB

El **MPLAB** (*Microchip Laboratory*) es el entorno de desarrollo que normalmente se usa para crear los programas que se graban en el interior de los Microcontroladores PIC. Este entorno es de distribución gratuita.

Microchip desarrolló un entorno de desarrollo muy flexible, económico y práctico el cual le permite al usuario crear un programa, simular su funcionamiento y descargar el programa dentro de un PIC a partir de una serie de programadores licenciados por la compañía. El MPLAB es la herramienta más completa del mercado ya que integra toda una serie de herramientas de software muy útiles para el desarrollador las cuales facilitan la tarea de diseño.

El MPLAB permite hacer **Debugger In Circuit**, es decir, depurar nuestro programa directamente sobre nuestro circuito electrónico controlando el programa desde el mismo MPLAB corriendo el programa paso a paso o por segmentos de código.

CREAR UN PROYECTO CON EL MPLAB

Para crear un programa en el MPLAB, realizaremos un **proyecto** y seguiremos los siguientes pasos:

- Pulsar el menú **Project**.
- Dentro de Project seleccionaremos Project Wizard. Una vez desplegado el mensaje de bienvenida pulsamos el botón que dice **Siguiente**.
- En el **step one** (**paso1**) se nos pide seleccionar el tipo de dispositivo que usaremos en la ventana **device**.
- En el **step two** (**paso 2**) se nos pide seleccionar el **Toolsuite** que consiste en el tipo de compilador, por default vienes el MPASM que es el Assembler, y pulsamos nuevamente **Siguiente**.
- En el **step three** (**paso 3**) se nos pide crear un proyecto nuevo o reconfigurar el actual, para crear uno nuevo pulsamos el botón **Browse...**. Se desplegará una ventana donde asignaremos un nombre a nuestro proyecto y lo guardaremos en el directorio **Microchip** pulsando el botón **guardar.** Esta acción provocará que volvamos al paso tres y pulsaremos el botón **Siguiente**.
- En el **step four** (**paso 4**) se nos pide que adjuntemos un archivo que hayamos creado, podemos tomar los archivos **template**, los cuales son plantillas creadas por Microchip para facilitar la escritura de un programa; existen tantos templates como modelos de microcontroladores. Para seleccionar uno de estos templates deberemos ingresar a la carpeta **Microchip** y dentro de esta entraremos a la carpeta **MPAM Suite.** Dentro de esta carpeta ingresaremos a **Template** y allí a la carpeta **Code.** En esta carpeta encontraremos los archivos template y seleccionaremos según el modelo del microcontrolador que usemos una de ellas haciendo clic con el mouse sobre esta. Luego pulsaremos el botón **Add**, lo que nos permitirá adicionar el archivo marcado a nuestro proyecto. El paso siguiente será pulsar el boton **Siguiente** lo que nos permitirá pasar a la siguiente ventana.

En la ventana **Summary** pulsaremos el boton **Finalizar** y tendremos creado nuestro primer proyecto, el cual aparecerá en nuestra ventana de trabajo como ventana de proyecto con el nombre y las carpetas de cada archivo, pegada a la carpeta **Source File** aparecerá nuestro archivo **template**.

LAS DIRECTIVAS

En los programas escritos en lenguaje Assembler existen las directivas que son órdenes al compilador y que no forman parte del set de instrucciones del PIC. La directiva le indica al compilador cómo realizar el proceso de traducción. **Figura 12**.

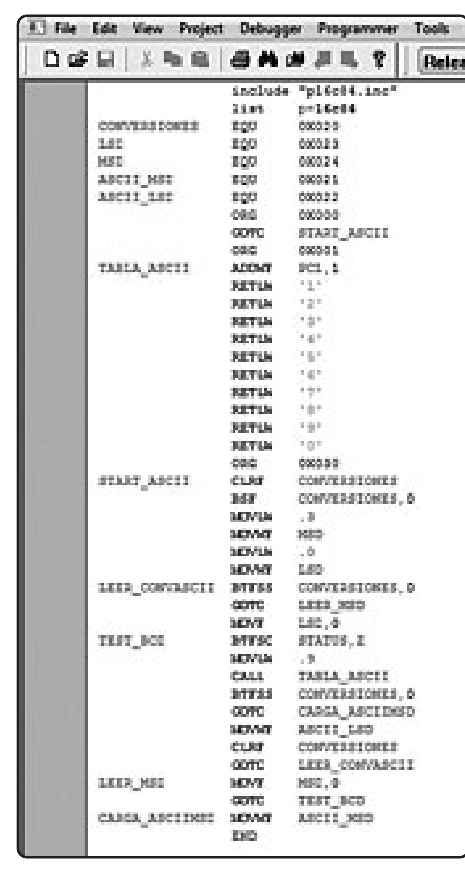
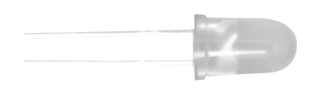


FIGURA 12. En la figura vemos el uso de las directivas.

Microchip desrrolló un entorno de desarrollo muy flexible, económico y práctico

El compilador MPASM tiene muchas directivas, sin embargo las mas utilizadas son las siguientes:

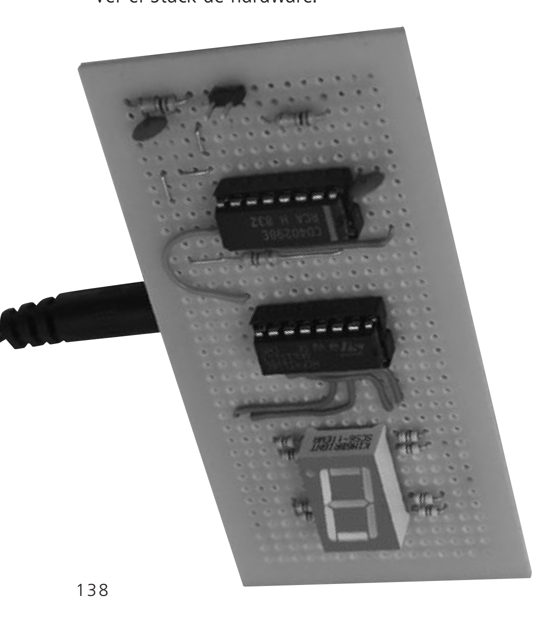
- **ORG**: le indica al compilador a partir de qué posición de memoria de programa se deberá cargar el código que se encuentra a continuación de esta dentro del PIC.
- **LIST**: con esta directiva le indicamos al compilador para qué modelo de microcontrolador hemos escrito el programa.
- **EQU**: nos permite establecer una equivalencia entre una etiqueta y un valor o posición de memoria.
- **INCLUDE**: nos permite incluir dentro de nuestro archivo un archivo externo, el cual tiene la extensión INC, por ejemplo los archivos de definiciones de los registros de funciones especiales.
- **END**: le dice al compilador que debe terminar el proceso de compilación en ese punto, se usa al final de todo programa en Assembler.
- **BANKEL**: nos permite seleccionar de forma automática un banco de registros, esta directiva genera el código Assembler necesario para modificar los bits RP0 y RP1 que controlan el cambio de página.



EL MPLABSIM

Una herramienta muy útil que integra el MPLAB es el MPLABSIM, que consiste en un simulador por software. Esta herramienta nos permite convertir a nuestra PC en un PIC virtual y poder correr el programa sobre este pudiendo observar el estado que adquiere el PIC. El MPLABSIM nos permitirá:

- Correr el programa paso a paso o hasta que encuentre una instrucción de alto (**BreakPoint**), o correr el programa en modo continuo.
- Detener el programa en cualquier momento.
- Ver el valor que adquieren los registros de la RAM.
- Ver el valor que adquieren los registros de propósitos especiales.
- Ver solo el estado de algunos registros (usando la **WatchWindow**).
- Si el PIC tiene EEPROM podremos ver su contenido.
- Ver el Stack de hardware.



Todas estas funciones podrán ser visualizadas activando cada una de las funciones que nos permite el simulador y se presentarán como ventanas que acomodaremos en nuestro ambiente de trabajo para poder visualizarlas.

En todo proceso de creación de un programa es bueno hacer la simulación para ver que aquel se comporte del modo esperado en nuestro diagrama de flujo.

Si bien el MPLABSIM nos permite aplicar estímulos externos para modificar el funcionamiento de nuestro programa debemos saber que el comportamiento está simulado y que nunca reemplaza a la prueba final sobre el prototipo. Esto es así para todos los simuladores pues en la realidad pueden aparecer eventos no esperados, consecuencia de variables independientes como ser:

- Diseño del PCB.
- Material del PCB.
- Longitud de las pistas del PCB.
- Efecto rebote de pulsadores.
- Ruido por emisiones electromagnéticas cercanas, etc.

Sin embargo el MPLABSIM es una herramienta excelente cuando queremos ver el funcionamiento interno del microcontrolador al correr nuestro programa.

Veremos en el **Paso a paso 1**, que se encuentra en la página siguiente, cómo hacer funcionar el **MPLABSIM** en un pequeño programa simple, el cual hará cambiar el estado de un puerto. Crearemos nuestro programa, que será muy sencillo ya que consiste en encender y apagar el estado del puerto **RB0** de un **PIC16F887**. (**Figura 13**).

PASO A PASO /1
Prácticas con el MPLABSIM

1



Para activar el **MPLABSIM** vaya hasta la barra de herramientas y hagas clic sobre `Debugger` y luego presione `Select_tool`, dentro del cual deberá seleccionar `MPLAB SIM`.

2



Para activar y detener el simulador usará los controles del simulador, este le permite correr el programa en forma continua, de forma animada, o paso a paso.

PASO A PASO /1 (cont.)

3



Lo primero que hará será resetear virtualmente el **PIC**, pulsando el boton de **RESET**, con lo cual el MPLABSIM pasará a ejecutar la primera instrucción. La instrucción a ejecutar se indica con una flecha.

4



Antes de Iniciar el funcionamiento, seleccionará los registros más importantes afectados por el programa, para poder monitorear su estado usando la función **watch**. Para ello vaya a la **barra de menues** y haga clic sobre **view**, seleccionando luego la herramienta **watch.**

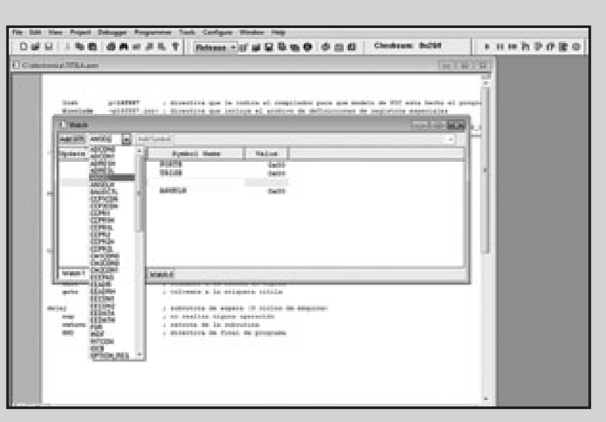
PASO A PASO /1 (cont.)

5



Seleccionará los distintos registros haciendo clic sobre el botón de selección y adicionará cada uno usando el botón **Add SFR**, para el ejemplo se han seleccionado **PORTB**,**TRISB**,**ANSEL** y **ANSELH**.

6



Minimice la **Watch windows** y arrástrela hasta posicionarla a un costado de la ventana de trabajo, para poder monitorear los cambios que se producirán a cada paso del programa observe la ventana de trabajo donde se encuentra el programa y a un costado la **Watch Windows** con los registros seleccionados

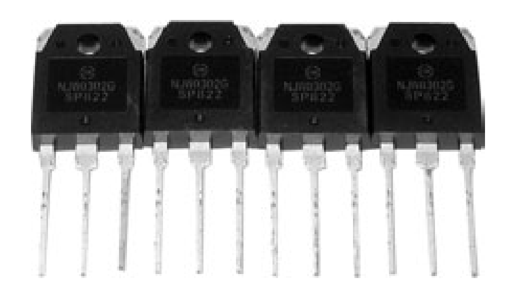
PASO A PASO /1 (cont.)





Para hacer funcionar el programa usará el boton **`Step Into`** que le permitirá ejecutar cada instrucción paso a paso, a su vez observe el estado que adquiere cada registro en la **`Watch Windows`**. Puede ver cómo se ejecuta paso a paso el programa y a su vez el resultado de la ejecución de cada instrucción en la **`Watch Windows`**.

El grabador **de PICs**



Para grabar los microcontroladores PIC usaremos el programador que armamos, con el este podremos programar cualquier PIC16F y PIC18F que tenga encapsulados DIP18 o DIP40.

La placa programadora que ya hemos armado la usaremos para introducir nuestro programa en la memoria de programa del PIC. A esta placa generalmente de la denomina **quemador**. Como nuestro programador casero no está licenciado por Microchip, el MPLAB no nos brinda soporte para este. Por ello, para controlar la placa programadora, usaremos un software que se obtiene libre en Internet y que se denomina WINPIC800, también existen otros muy buenos como el ICPROG y el WINPICPGM.

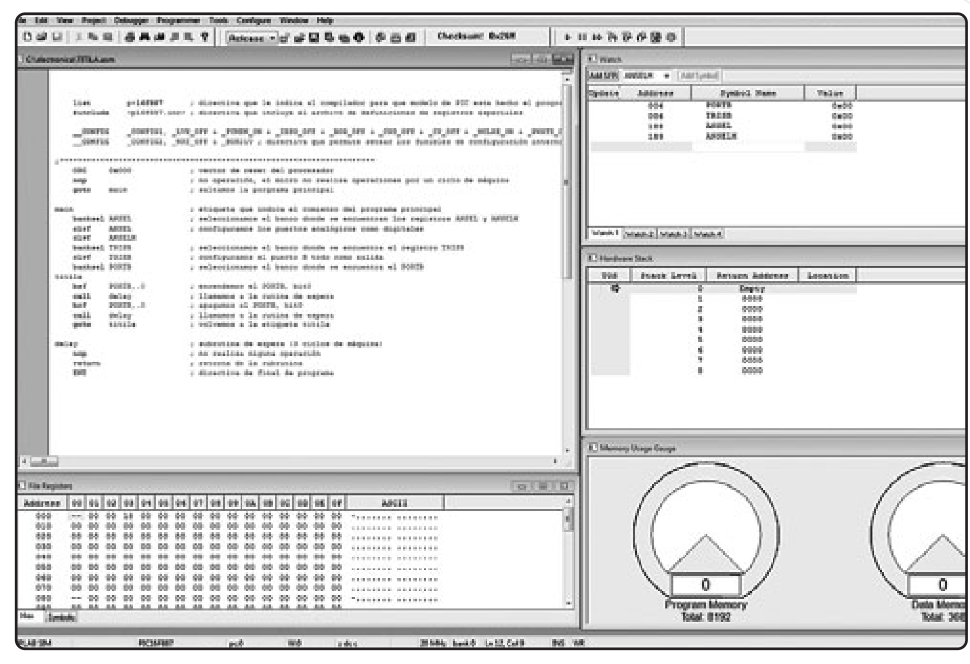


FIGURA 13. **En la figura vemos el MPLABSIM en acción.**

CONFIGURANDO EL PROGRAMADOR

Lo primero que tenemos que hacer es configurar el hardware y el software. Como este software controla una gran cantidad de programadores deberemos setear el nuestro. Para ello iremos al menú **Configuración** y seleccionaremos **Hardware.**

Una vez que seleccionemos **Hardware**, en la ventana que se despliega elegiremos el programador **JDM Programmer** y configuraremos el puerto **COM**, donde se encuentre conectada nuestra placa. Después de toda esta operación ya queda configurado el software del WINPIC800 para controlar nuestro programador.

GRABACIÓN, LECTURA Y BORRADO DE UN PIC

El proceso de grabación de un PIC es muy simple, solo tenemos que cargar el archivo HEX (resultado de la compilación) para lo cual iremos al icono **`Abrir`** y seleccionaremos nuetro archivo. Una vez cargado podremos grabarlo usando el ícono **`Programar Todo`**, para borrar el PIC usaremos el icono **`Borrar Todo`** y para leer el contenido del PIC usaremos **`Leer Todo`**.

Multiple **choice**

1 **¿Cuántos pines posee el encapsulado del microcontrolador PIC16F887?**

a- 2.
b- 8.
c- 35.
d- 40.

---

2 **¿Cuántos pines de entrada/salida tiene el microcontrolador PIC16F887?**

a- 2.
b- 8.
c- 35.
d- 40.

---

3 **¿Cuántos modos diferentes posee el oscilador del microcontrolador PIC16F887?**

a- 2.
b- 8.
c- 35.
d- 40.

---

4 **¿Cuántos osciladores internos posee el microcontrolador PIC16F887?**

a- 2.
b- 8.
c- 35.
d- 40.

---

5 **¿Cuántos bits tiene el bus de datos del microcontrolador PIC16F887?**

a- 2.
b- 4.
c- 8.
d- 16.

---

6 **¿Cuántos tipos de memorias posee el microcontrolador PIC16F887?**

a- 2.
b- 4.
c- 8.
d- 16.

Respuestas: 1 d, 2 c, 3 b, 4 a, 5 c, 6 a.

Capítulo 6

Microcontrolador PIC18F



Abordaremos las características principales y analizaremos la estructura interna de los PIC18F.

Microcontroladores **PIC18F**

Los PIC18F son los sucesores de la familia PIC16F. Incorporan una gran cantidad de cambios que han transformado al PIC en el mejor microcontrolador de 8 bits.

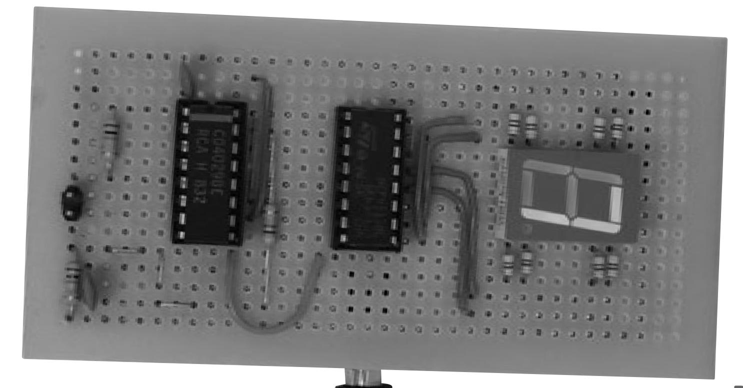
En 1999 Microchip lanzó al mercado una nueva familia de microcontroladores, bautizada como **PIC18**. Incorporaba una renovación dentro del núcleo del microcontrolador que tendía a facilitarle al desarrollador el trabajo de programación.

PIC18 se pensó para ser programado en lenguajes de alto nivel como C, por lo cual Microchip dotó a su nueva familia de las siguientes mejoras dentro del núcleo: memoria de programa de hasta **1 MWord** (1 millón de instrucciones), acceso lineal a la memoria de programa, memoria de datos de hasta 4 KB y acceso a memoria de datos lineal o por bancos. Además, cuenta con un set de instrucciones ampliado a 75 en modo estándar y a 83 en modo extendido (en los modelos que lo incorporan), entre otras características.

Características **de la familia PIC18F**

Estas microcomputadoras son fabricadas por la firma Microchip y se ofrecen con una amplia gama de funcionalidades, destacándose por su sencillez y bajo consumo.

Los microcontroladores **PIC18F** pertenecen a la familia de rango medio avanzado de los micros de 8 bits de Microchip. Poseen un repertorio de 75 instrucciones en modo estándar, más 8 instrucciones especiales que solo trabajan cuando el microcontrolador opera en modo extendido (con memoria de programa interna y externa). El procesador sigue siendo de filosofía **RISC** (set de instrucciones reducido), buses de arquitectura **Harvard,** construido en pastilla de tecnología **CMOS**, frecuencia de operación de hasta **40 MHz**, hasta **32 KWord** de memoria **Flash** de programa, conversores A/D integrados y manejo de interrupciones. Son ideales para aprender las bases de la programación de estos dispositivos.



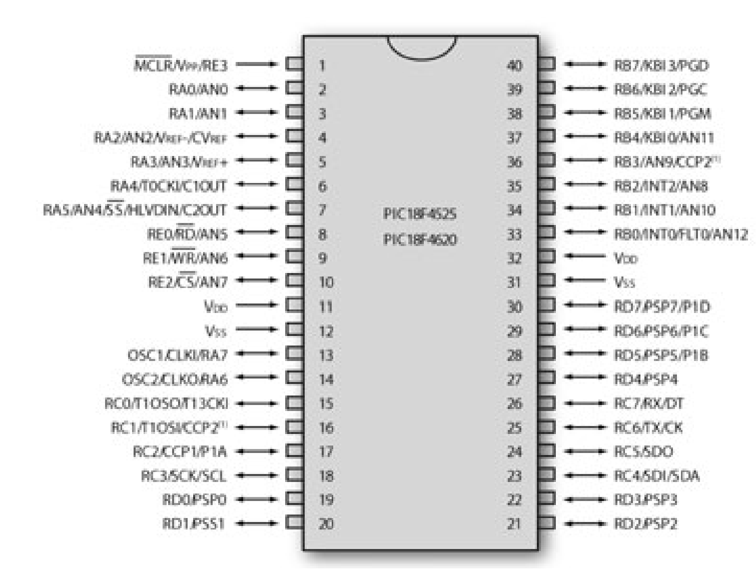


FIGURA 1. **Aquí vemos la disposición de los pines del microcontrolador y sus funciones.**

Su nuevo set de instrucciones, que ampliaba la capacidad de las **35** de la familia PIC16, le permitía al programador de lenguaje assembler una mayor comodidad al realizar el código. Pero uno de los detalles más importantes es que PIC18 se pensó para ser programado en lenguajes de alto nivel, como C, y fue por eso que Microchip desarrolló el hardware a la vez que lo hacía con el compilador, para amoldarlo a las características de este último, al cual denominó C18. Así, PIC 18 se transformó en el primer microcontrolador diseñado para ser programado en lenguaje C.

MICROCONTROLADOR PIC18F4620

Posee un encapsulado de 40 pines del tipo **DIP** (*Dual In-Line Package*), cuya disposición podemos observar en la **Figura 1**. El PIC 16F4620 se destaca del resto de los que integran la familia por tener:

- 32 KWord de memoria de programa
- 3968 bytes de memoria RAM
- 1024 bytes de memoria de datos en EEPROM
- 13 canales A/D, el cual es de 8 bits
- 1 timer de 8 bits y 3 de 16 bits



UN DATO IMPORTANTE

Una de las novedades que presenta PIC18F con respecto a PIC16 está el salvado de contexto automático para las interrupciones. En él, el PIC18F salva el contenido de los registros WREG, BSR y STATUS sobre los registros sombra.

TIPOS DE OSCILADOR

El **PIC16F4620** tiene 10 modos distintos de oscilador, que se seleccionan mediante la palabra de configuración. Hay que programar los bits de configuración **FOSC3**: **FOSC0** para seleccionar un modo, el cual puede ser:

- **LP**: Cristal de baja potencia
- **XT**: XTAL/ resonador
- **HS**: XTAL/ resonador de alta velocidad
- **HSPLL**: XTAL/resonador de alta velocidad con PLL activo
- **RC**: capacitor/resistor con salida se FOSC/4 sobre RA6
- **RCIO**: capacitor/resistor con Port I/O sobre RA6
- **EC**: reloj externo con salida FOSC/4 sobre RA6
- **ECIO**: reloj externo con Port I/O sobre RA6
- **INTIO1**: oscilador interno con salida FOSC/4 sobre RA6 y Port I/O sobre RA7
- **INTIO2**: oscilador interno con Port I/O sobre RA6 y RA7

Esta familia de dispositivos incluye un circuito PLL (Phase Locked Loop) o lazo enganchado de fase

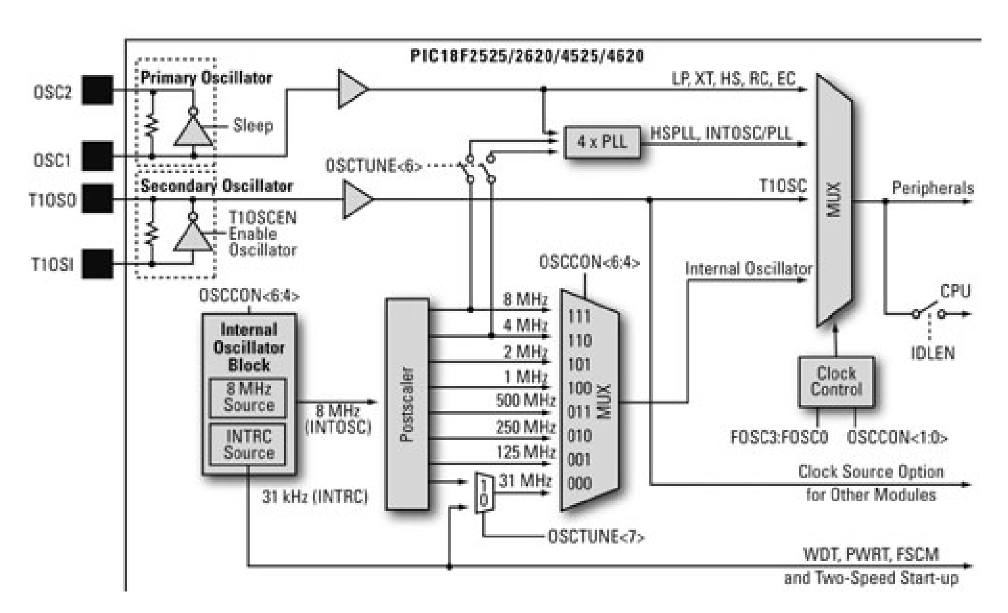


FIGURA 2. Vemos el diagrama del oscilador interno del PIC18F4620.

Recordemos que los bits de configuración nos permiten establecer las características especiales del microcontrolador: tipo de oscilador, protección contra lectura de la memoria de programa y watchdog timer, entre otras.

EL OSCILADOR INTERNO

Los **PIC18F4620** tienen un oscilador interno que genera dos señales diferentes, cualquiera de las cuales puede utilizarse como reloj del microcontrolador. Este componente puede eliminar el oscilador externo de los pines **OSC1** y/o **OSC2**.

La salida principal (**INTOSC**) es una fuente de reloj de **8 MHz** que puede emplearse para dirigir el reloj directamente. Esto también gobierna el **postscaler** o divisor de frecuencia programable del INTOSC, el cual puede proporcionar un rango de frecuencias de **31 KHz** a **4 MHz**. La salida **INTOSC** está activa cuando se selecciona una frecuencia de reloj de **125 KHz** a **8 MHz.** (**Figura 2**)

EL PLL INTERNO

Esta familia de dispositivos incluye un circuito **PLL** (*Phase Locked Loop*) o lazo enganchado de fase. Esta opción puede usarse para que, a partir de un oscilador de baja frecuencia de clock, se obtenga una alta frecuencia de operación interna. Esto es muy útil para reducir las interferencias tipo **EMI** que se generan en el uso de cristales de alta frecuencia de operación. En el modo **HSPLL (oscilador PLL de alta velocidad**) es posible trabajar con un cristal externo de hasta **10 MHz**.

CIRCUITO DE RESET

Cuando se produce un reset, el procesador se reinicializa, el programa en ejecución se abandona y el contador de programa se carga con la **dirección 0**. Es allí donde comienza el programa de la aplicación (**Figura 3**).Las dos formas más comunes para provocar un reset son:

- Quitar y volver a establecer la alimentación del microcontrolador. Cuando se conecta la alimentación, se produce automáticamente un reset, conocido como **Power-On Reset** (POR).
- Aplicar un nivel lógico bajo en el terminal **MCLR** (*Master Clear Reset*), que permite provocar un reset externamente en el momento en que queramos, sin necesidad de desconectar el microcontrolador de su alimentación. Para introducir un nivel bajo en este pin, debemos colocar un pulsador que lo lleve a ese estado. En la **Figura 3**, que se encuentra en la siguiente página, tenemos la posibilidad de observar un esquema típico para el circuito de reset de un microcontrolador.



MEJORAS TECNOLÓGICAS

Microchip mejoró el hardware de los puertos al colocar dos registros: uno es el denominado **PORT**, para manejar la entrada de datos, y el otro es el **LAT**, para controlar la salida. Esto solucionó el problema que tenía la familia PIC16 en la reconfiguración del puerto.

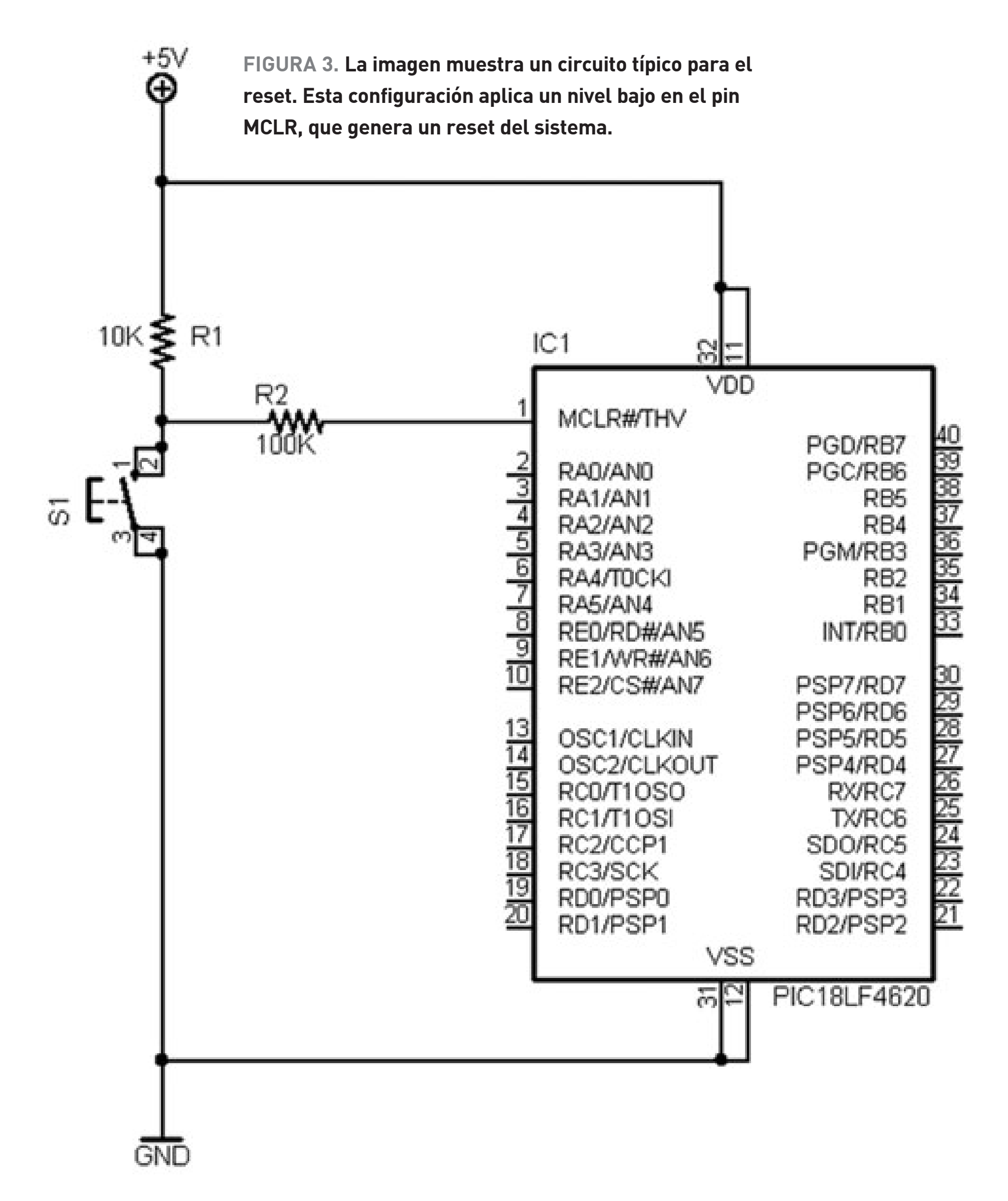


FIGURA 3. La imagen muestra un circuito típico para el reset. Esta configuración aplica un nivel bajo en el pin MCLR, que genera un reset del sistema.

PUERTOS DE ENTRADA Y SALIDA

El microcontrolador **PIC18F4620** posee cinco puertos de entrada y salida: **A, B, C, D y E**. Debido a que el bus de datos es de **8 bits**, es lógico que el tamaño de los puertos sea el mismo. A diferencia de otros microcontroladores PIC, los puertos **A, B, C** y **D** son de **8 bits,** mientras que el **E** es de **4 bits**. Cada pin de cada puerto puede configurarse de manera individual como entrada o salida digital para recibir o enviar datos, según las necesidades, a excepción del **RE3**, que solo puede funcionar como entrada.

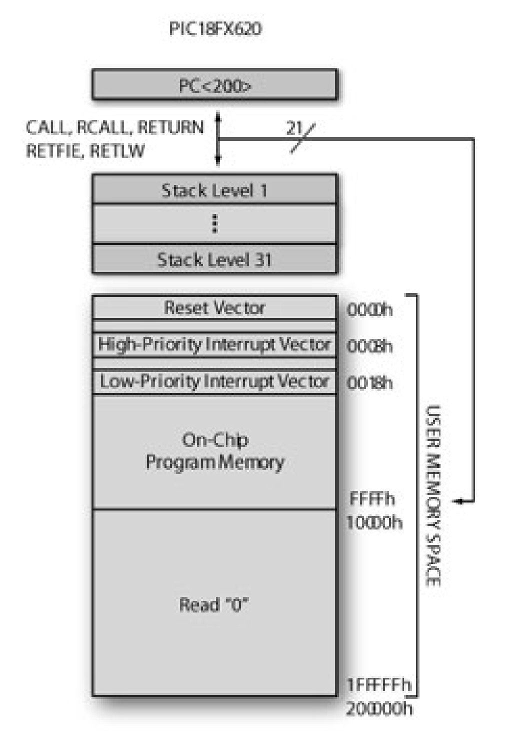


FIGURA 4. La memoria de programa tiene 32 KWord con direccionamiento completamente lineal, salvando las limitaciones que tenían los PIC16F y su sistema paginado.

Pero no todos los puertos son solo **E/S digitales**. Para ahorrar pines en el encapsulado, estos se comparten entre los distintos periféricos. Por ejemplo, el puerto **A** y parte del **B** pueden usarse como entrada analógica para el conversor A/D incluido en el integrado. Por tanto, el **PIC18F4620** tiene un total de **35 puertos I/O**, más uno que únicamente puede operar como entrada. Sin embargo, de usarse toda la capacidad I/O, deberá emplearse el oscilador interno.

ORGANIZACIÓN DE LA MEMORIA

Hay tres tipos de memoria en el **PIC18F4620**: de programa, RAM de datos y EEPROM de datos.
La primera almacena el programa que controlará el funcionamiento del PIC, y es la que grabamos normalmente con el programador de PICs durante el proceso de programación. En la memoria **RAM** se almacenan los datos manejados por el programa que cargamos en el PIC y los del propio micro. A esta última se la denomina **RAM del sistema** o **registros de funciones especiales**.

Por su parte, la memoria **EEPROM** de datos es usada también para contener datos, pero a diferencia de la RAM, no necesita alimentación para retener la información, pues es no volátil. Suele usarse para realizar backups.

LA MEMORIA DE PROGRAMA

Los microcontroladores **PIC18F** implementan un contador de programa de 21 bits que es capaz de tratar 2 MB de memoria de programa. Esta memoria

puede ser tratada como memoria de **16 bits** cuando se leen instrucciones, o como bytes cuando se **leen datos**. Es posible armar tablas de datos a todo lo largo de ella, y estas se encontrarán limitadas solo por la memoria física del PIC.

A diferencia de las familias anteriores, los **PIC 18** no tienen un esquema de memoria de programa segmentada en páginas, sino que sus direcciones son lineales, debido a la ampliación de la capacidad del contador de programa. Teóricamente, dentro de los PIC18 es posible implementar una memoria de programa de hasta **1 MWord**, enorme aun para los compiladores de alto nivel.

En este nuevo esquema, las subrutinas pueden ser llamadas desde cualquier lugar de la memoria de programa, y es posible realizar un salto dentro del programa hacia cualquier parte de él sin preocuparnos por la extensión que este deba tener (**Figura 4**).



CONTADOR DE PROGRAMA

El **contador de programa** (PC) especifica la dirección de la instrucción que se va a ejecutar. Tiene **21 bits** de ancho, separados en **tres registros de 8 bits**. El byte bajo es conocido como registro **PCL**. El byte alto, o registro **PCH**, contiene los bits <15:8> del PC. Las actualizaciones del registro PCH se realizan a través del registro de **PCLATH**. El tercer registro o **byte superior** se llama **PCU**, y contiene los bits <20:16> (**Figura 5**).

Las actualizaciones del registro PCU se realizan a través del registro **PCLATU**.

El contenido de PCLATH y de PCLATU se transfiere al contador de programa por cualquier operación que escriba el **PCL**. De la misma manera, los dos bytes superiores del contador de programa se transfieren al PCLATH y al PCLATU por cualquier operación que lea el PCL.

El contador de programa (PC) especifica la dirección de la instrucción que se va a ejecutar



PIC18F

PIC18F ha servido de base para todos los microcontroladores posteriores. Muchas de las características de los PIC18 fueron utilizadas para reformular los **PIC16F** y generar la nueva familia de **rango medio mejorado** o **PIC16F1XXX**, presentada por Microchip.

Contador de Programa o PC

PCLATU	—	—	—	PC<20:16>
PCLATH	PC<15:8>			
PCL	PC<7:0>			

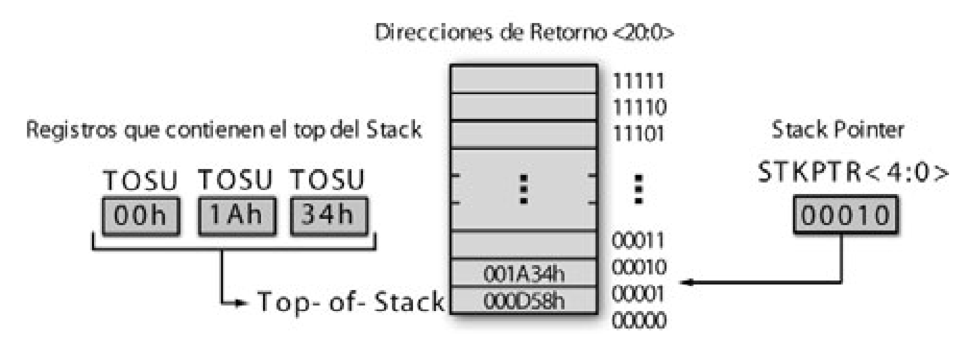


FIGURA 5. Contador de programa y pila o stack de 31 niveles. El contador siempre tiene la dirección de la próxima instrucción por ejecutar.

MEMORIA DE DATOS

La memoria de datos en los dispositivos **PIC18** se implementa como **RAM** estática. Cada registro en la memoria de datos tiene una dirección de **12 bits**, lo que permite hasta **4096 bytes** de memoria. Se divide en **16 bancos** con **256 bytes** cada uno. Los dispositivos **PIC18F4620** tienen **16** bancos completos, para un total de **4096 bytes** (**Figura 6**).

La memoria de datos contiene los registros de función especial (SFRs) y los registros de funciones generales (GPRs). Los primeros se utilizan para el control y el estado del regulador y de las funciones periféricas, mientras que los segundos se emplean para el almacenamiento de datos y las operaciones destinadas a guardar datos del usuario. Cuando se lee una localización no implementada, aparecen ceros.

La memoria de datos en los dispositivos PIC18 se implementa como RAM estática

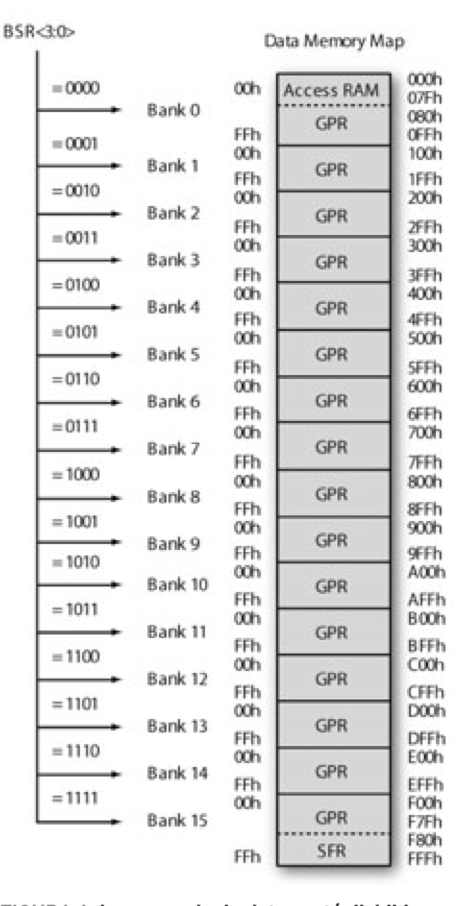
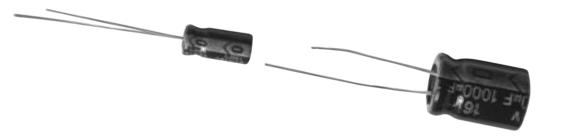


FIGURA 6. **La memoria de datos está dividida en 16 bancos de 256 posiciones cada uno. Los registros de funciones especiales se agrupan en el último banco (banco 15).**



ACCESS BANK

El sistema y la arquitectura de las instrucciones permiten realizar operaciones a través de todos los bancos. Es posible tener acceso a la memoria entera de datos por modos de direccionamiento directo, indirecto o puestos en un índice.

Para asegurarse de que los registros de uso general (SFRs y GPRs) se alcancen en un solo ciclo, los dispositivos **PIC18** tienen un banco de acceso directo denominado **acces bank**. Se trata de una memoria de **256 bytes** que proporciona el acceso rápido a SFRs y a la parte más baja del banco 0 de GPR sin usar BSR.

REGISTROS DE PROPÓSITO GENERAL

En los dispositivos PIC18, es posible almacenar datos en el área **GPR**. Esta es la **RAM** de datos disponible para el uso de todas las instrucciones. GPR empieza en el fondo del banco 0 (**dirección 000h**) y crece hasta la última área de **SFR**. No se inicializa por un reset y no cambia en ningún otro.

REGISTROS DE FUNCIONES ESPECIALES

Los registros de función específica (SFR) son usados por la **CPU** y por los módulos periféricos para controlar el funcionamiento del dispositivo. Están implementados como **RAM estática** en la memoria de datos. Los **SFR** empiezan en la última dirección de memoria de datos y se extienden hacia abajo hasta ocupar el segmento superior del banco **15**, de **F80h a FFFh**. Pueden clasificarse en dos grupos: los asociados a la funcionalidad del núcleo del dispositivo y los relacionados con las funciones de los periféricos.



FIGURA 7. En la figura observamos los distintos bits que forman el registro STATUS. El esquema es similar al del PIC16F, pero aquí se suman el bit N (negativo) y el OV (overflow).

EL REGISTRO DE ESTADO

El registro **STATUS** contiene el estado aritmético de la ALU. En su interior podemos encontrar distintos bits que indican el estado de un producto aritmético o lógico. Entonces, como en los **PIC16F**, encontramos el bit de cero **Z**, que se pone en **uno** cuando el resultado es **cero**, y los bits de acarreo, tanto para el bit más significativo, como el bit intermedio. Pero, a diferencia de los PIC16, los PIC18 agregan el bit de overflow (**OV**, desbordamiento), que se activa cuando se produce un desbordamiento en el resultado de la cuenta. También agregan el bit Negative (**N**, negativo), cuando el resultado de la operación es negativo.Como con cualquier otro **SFR**, es posible modificarlo con cualquier instrucción.

Si el registro STATUS es el destino de una instrucción que afecta a los bits **Z**, **DC**, **C**, **OV** o **N**, no escribirá el resultado de la instrucción. En cambio, se actualiza según la instrucción ejecutada. Por lo tanto, el resultado de una instrucción con el registro STATUS como destino puede ser diferente de lo previsto. Por ejemplo, **CLRF STATUS** activará el bit **Z,** y el resto de los bits no cambiará (**Figura 7**).

Los SFR controlan los puertos de entrada/salida y la operación de los periféricos internos

EL REGISTRO BSR

Su sigla proviene de *Bank Select Register* (registro selector de bancos). La mayoría de las instrucciones en el sistema del PIC18 hacen uso del puntero de banco, conocido como BSR. Este SFR controla los 4 bits más significativos de la dirección de



DIFERENCIAS ENTRE PIC18 Y PIC16

A diferencia de PIC16, PIC18 tiene todos los registros de funciones especiales agrupados en las últimas 128 posiciones de la página 15 de la memoria de datos, lo cual le permite al usuario acceder a ellos de forma directa sin tener que cambiar siempre de bancos.

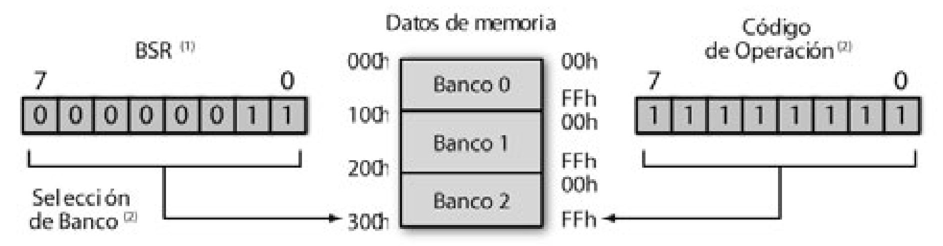


FIGURA 8. El registro BSR mediante el cual podemos acceder a cualquier banco de la memoria RAM.

localización; la instrucción incluye los 8 bits menos significativos. Solo los 4 bits más bajos del BSR están implementados (BSR3:BSR0). No se utilizan los 4 altos, que leerán **0** y no se pueden escribir. El BSR se puede cargar directamente usando la instrucción **MOVLB**.

El valor del BSR indica el banco en la memoria de datos. Los 8 bits de la instrucción muestran la localización dentro del banco (**Figura 8**).

ACCESO A LOS BANCOS

Mientras que el uso de **BSR**, con una dirección de **8 bits**, permite que los usuarios traten la gama entera de memoria de datos, también significa que siempre hay que asegurarse de que esté seleccionado el banco correcto. De no ser así, los datos se leerán o escribirán en una localización incorrecta. Puede ser problemático si un **GPR** va a ser modificado por una operación, porque se escribirá un **SFR** en vez de otro (**Figura 9**).

Verificar o cambiar el BSR para cada lectura o escritura en la memoria de datos puede resultar in-

Para mejorar el acceso a las posiciones de memoria de datos de uso general, esta se configura con un banco virtual



BANCO VIRTUAL

El banco virtual es muy útil porque simplifica la programación. Sin embargo, hay que tomar en cuenta que solo tenemos 128 registros de propósitos generales para almacenar nuestros datos, ya que los otros 128 están ocupados por los registros de funciones especiales.

eficiente. Para mejorar el acceso a las posiciones de memoria de datos de uso general, esta se configura con un banco virtual, que permite a los usuarios acceder a un bloque mapeado de la memoria sin especificar un BSR. El banco virtual consiste en los primeros **128 bytes** de memoria (**00h-7Fh**) en el banco **0** y los últimos **128 bytes** de memoria (**80h-FFh**) en el **bloque 15**. La mitad inferior se conoce como **acceso RAM** y se compone de los **GPRs**. La mitad superior es donde los SFRs del dispositivo están mapeados. A estas dos áreas mapeadas en el banco virtual se puede acceder con una dirección de **8 bits**. Este banco virtual está activo por default en el momento de arranque del **PIC18F4620**.

De esta forma, la programación desde assembler es muy sencilla, porque el programador no debe estar utilizando cambios de bancos como sucedía con el **PIC16F**. En este esquema, la memoria **RAM** puede ser considerada como una memoria de datos de direccionamiento lineal. Este concepto está tomado de los microprocesadores viejos como el **R6502,** que tenían direccionamiento en página cero, lo cual simplificaba el acceso a la memoria.

MPLAB C18

Dentro de este entorno de desarrollo podemos trabajar con los PIC18F y programarlos en lenguajes de alto nivel, como C. Veamos cómo realizar estos procesos. Para la programación de microcontroladores es importante elegir cuál será el lenguaje por utilizar. La elección estará definida por distintos factores, como la velocidad deseada en tiempos de programación, el tamaño del código y la facilidad de programación.

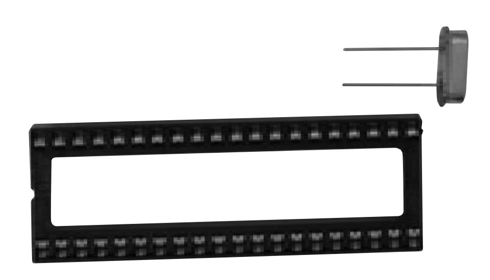


FIGURA 9. Aquí vemos el modo banco virtual en funcionamiento, donde se agrupan los primeros 128 registros del banco 0 con los últimos 128 del banco F.



MÁS SOBRE PIC18

El microcontrolador PIC18 supera ampliamente a PIC16, pues el programador ya no tiene que estar agregando en su código los cambios de página. Los SFRs y los GPRs se encuentran en el mismo banco virtual.



En la actualidad, existen varios compiladores de C para los microcontroladores PIC

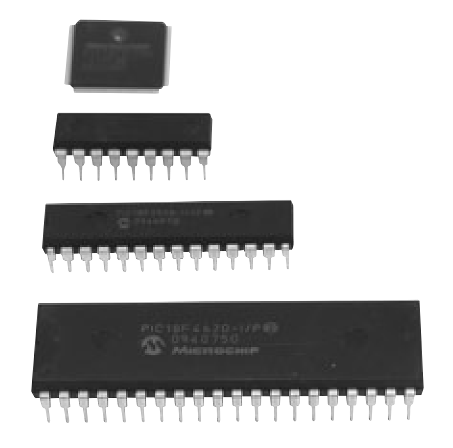
Uno de los motivos más importantes para escoger un lenguaje de alto nivel en vez del ensamblador, es cuando la complejidad del programa aumenta considerablemente. En ese caso, se prefiere utilizar un lenguaje que se acerque al humano, porque es más fácil de programar, produce código más legible y libera al programador de lidiar con los detalles del hardware, ya que el compilador se ocupa de toda esa tarea.

Dentro de todos los lenguajes posibles, se encuentra uno que es muy utilizado por los profesionales: **C**. Esto se debe a que, normalmente, es mas fácil escribir código C en vez de ensamblador, y dentro de los lenguajes de alto nivel, C es el de mas bajo nivel, lo que hace que sea soportado por distintos compiladores para microcontroladores con diversas arquitecturas.

FIGURA 10. **Este es el flujo de las herramientas de desarrollo para compilar un archivo escrito en lenguaje C, los distintos archivos implicados en la compilación y el ensamblaje a código de máquina para el PIC.**

COMPILADORES C PARA PIC

En la actualidad, existen varios compiladores de C para los microcontroladores PIC, los cuales ofrecen funciones similares. Entre los distintos compiladores que podemos encontrar en el mercado, los más destacados son los fabricados por **CCS**, **HiTech** y **Microchip**.



El programa **CCS C Compiler** es un compilador muy potente que soporta todas las familias de PICs y posee su propio entorno de trabajo, desde el cual podemos generar proyectos, simularlos y depurarlos. Este compilador cuenta, además, con una amplia librería de funciones (**Figura 10**).

Por su parte, **HiTech C Compiler** para PIC18 dispone de un plugin para incorporar en el entorno del **MPLAB IDE**. Si bien es menos conocido, no deja de ser una alternativa viable.

COMPILADOR MPLAB C18

El **MPLAB C18** es el compilador de Microchip para sus **PIC18** y tiene un lenguaje bastante similar al **C** convencional, excepto que se le han agregado diversas adaptaciones para volverlo más apropiado para el ambiente de programación de los PIC. Se maneja perfectamente dentro del entorno visual de MPLAB, y sus características son:

TIPO	TAMAÑO	MÍNIMO	MÁXIMO
Char	8 bits	-128	127
Signed Char	8 bits	-128	127
Unsigned Char	8 bits	0	255
Int	16 bits	-32768	32767
Unsigned Int	16 bits	0	65535
Short	16 bits	-32768	32767
Unsigned Short	16 bits	0	65535
Short Long	24 bits	-8.388.608	8.388.607
Unsigned Short Long	24 bits	0	16.777.215
Long	32 bits	-2.147.483.648	2.147.483.647
Unsigned Long	32 bits	0	4.294.967.295

TABLA 1. **Tipos de datos enteros del MPLAB C18. El compilador permite realizar operaciones matemáticas entre datos tipo char; el resultado también puede ser un char.**

- Solo se utiliza para los PIC18.
- Posee las funciones estándar del lenguaje C.
- Permite la inclusión de lenguaje ensamblador.
- Está optimizado para la arquitectura de los PIC18.
- Contiene librerías para comunicaciones **SPI**, **I2C**, **USART** y periféricos externos como **LCD** inteligentes.
- Ofrece una versión estudiantil gratuita.

Los compiladores de C para PIC más importantes son los fabricados por CCS, HiTech y Microchip

Todas estas características hacen del compilador **MPLAB C18** una herramienta ideal para los desarrollos que podemos realizar con los PIC18 (**Figura 11**).

TIPO DE ALMACENAMIENTO DE DATOS

El **lenguaje C** maneja los tipos de datos básicos, como **int**, **char** y **float**. El lenguaje orientado a los microcontroladores del MPLAB **C18** también los soporta. En la **Tabla 1** podemos ver los tamaños en bits que ocupan en la memoria. Esto es importante porque el tamaño de las variables puede cambiar de un compilador a otro, de modo que siempre que se escoge un compilador determinado para la programación, es conveniente verificar los tamaños de los tipos de datos que se utilizarán.

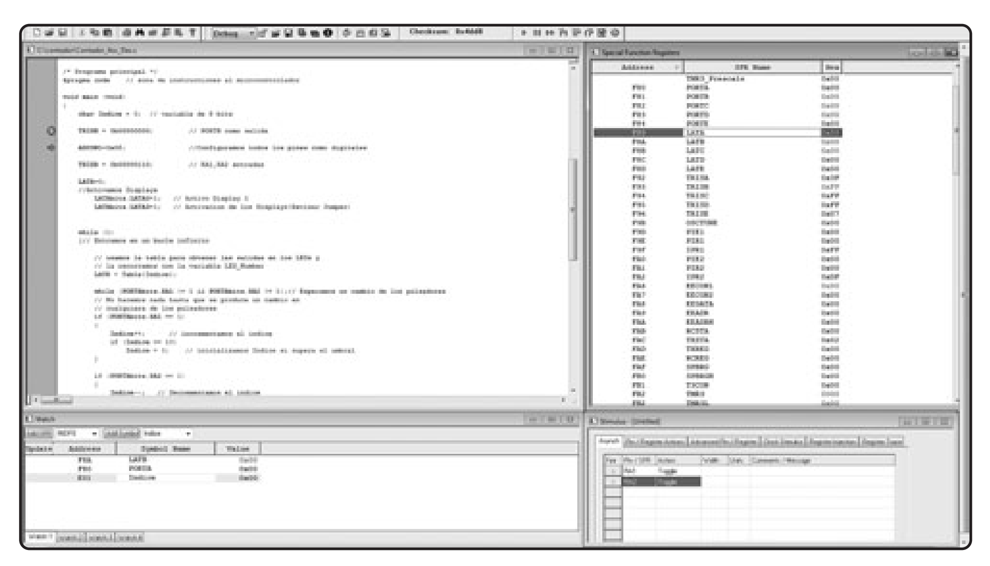


FIGURA 11. Aspecto del entorno de MPLAB cuando trabajamos en lenguaje C. Este software nos permite tener varias ventanas abiertas para seguir el estado del PIC en las simulaciones.

Los tipos de datos **char** almacenan un carácter simple, tienen un tamaño de **8 bits** y son ideales para trabajar con los registros del PIC, ya que poseen la misma longitud.

Las variables **int** tienen un tamaño de **16 bits**; son adecuadas para operaciones aritméticas con números enteros. Por su parte, para operaciones con mayor precisión tenemos las variables **float**, con un tamaño de 32 bits.

CADENAS DE STRING EN MEMORIA ROM Y RAM

Ya vimos los distintos tipos de datos que podemos manejar en el lenguaje, de modo que ahora nos concentraremos en las cadenas de **string** o caracteres. Esto nos permite manejar frases o mensajes en nuestro PIC con todas las funcionalidades que posee el **lenguaje C**.

Recordemos que en **C**, un **string** es un caso especial de **array**, y que en la declaración puede asignarse o no una dimensión del **string**. Si se opta por **no** asignar una dimensión, recordemos que el array de strings siempre finaliza con un **carácter nulo**.
La cadena de string puede ser almacenada en la memoria de programa (**ROM**) o en la de datos (**RAM**). El lugar donde se la guarda se indica con los calificadores **rom** o **ram**.





Veamos un ejemplo de cómo almacenar un string constante en la memoria de datos:

Const rom unsigned char micadena[] = "hola mundo"

Este ejemplo crea una cadena de caracteres constantes en la memoria **ROM**, y si bien no indicamos una longitud, sabemos que el último carácter será un **null**. Del mismo modo en que utilizamos los calificadores **far** y **near** para variables, también podemos usarlos para crear cadenas de caracteres en las posiciones de memoria que queramos.

ESTRUCTURAS Y UNIONES

Si queremos agrupar variables de diferentes tipos, podemos utilizar las **estructuras** y las **uniones**. El compilador **C18** permite declarar estructuras **anónimas** dentro de las uniones. La utilidad principal de esta característica es la declaración de variables (**Figura 12** y **13**). Por ejemplo:

```
Union tpuerto{
Unsigned char valor
struct{
          unsigned bit0:1;
          unsigned bit1:1;
          unsigned bit2:1;
          unsigned bit3:1;
          unsigned bit4:1;
          unsigned bit5:1;
          unsigned bit6:1;
          unsigned bit7:1;
};
}byte;
```

Esta es la declaración de una unión que, dentro, contiene una **estructura anónima**, porque no posee nombre. Esto permite utilizar la variable en el plano de los bytes y también a nivel de bits. Esto es similar al estándar en C para los campos de bits (*bitfields*), pero el hecho de que la estructura esté dentro de la unión permite modificar la variable, tanto en el nivel de los bytes como para acceder a los bits. Entonces, si queremos escribir un valor en el byte, utilizamos la variable de la siguiente forma:

```
// byte
byte.valor = 0xFF;
```

Del mismo modo podemos acceder a los bits de esa variable:

```
// bit
byte.bit0 = 0;
byte.bit7 = byte.bit0;
```

Este recurso es muy utilizado para definir los registros que maneja el microcontrolador.

Si queremos agrupar variables de diferentes tipos, podemos utilizar las estructuras y las uniones

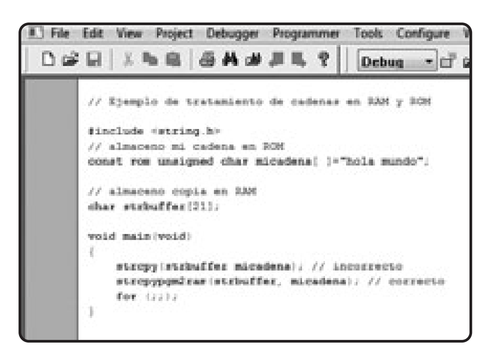


FIGURA 12. **Vemos cómo declarar cadenas de string en memoria de programa y almacenarlas en un buffer en la RAM.**



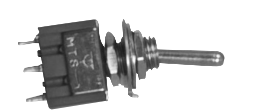
FIGURA 13. **Vemos cómo el compilador utiliza las uniones y las estructuras para definir los registros especiales (SFR) del PIC. En este extracto del archivo p18f4620.h vemos de qué modo se definen los bits del PORTA.**

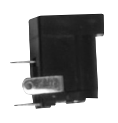
PUNTEROS DE MEMORIA

Los punteros son variables que contienen la dirección a una zona de memoria donde reside un determinado tipo de dato, (**Tabla 2**). En el compilador C18 podemos tener punteros hacia la memoria de datos o hacia la de programa. Un puntero a la memoria de datos tiene una longitud de **16 bits** y se declara de la siguiente manera: **char *puntero.**

Este puntero apunta a variables del tipo **char** en la memoria de datos. Un puntero a la memoria de datos no puede apuntar a un dato que se encuentre en la memoria de programa; para eso, debemos definir un puntero hacia esta memoria, por ejemplo: **near rom char *rom_puntero.**

Este contiene direcciones de datos que están en la memoria de programa y solo tiene acceso a las direcciones de memoria inferiores a 64 K. Si queremos usar punteros para direcciones mayores, debemos cambiar el prefijo **near** por **far**.





DIRECTIVAS

Al igual que el lenguaje C, el compilador C18 posee directivas al procesador que permiten realizar el control del código. La directiva **#pragma** permite a los compiladores definir sus propias directivas. El compilador C18 la utiliza para definir algunas secciones de memoria.

El compilador C18 divide las secciones de memoria en memoria de programa o memoria de datos. Estas secciones están subdivididas en dos zonas memoria de programa y memoria de datos.:

Memoria de programa
- **code**: instrucciones en memoria de programa.
- **romdata**: datos en memoria de programa.

Memoria de datos
- **udata**: variables de usuario estáticas sin inicializar.
- **idata**: variable de usuario estáticas inicializadas.

Para definir las rutinas de interrupciones, debemos utilizar la directiva **#pragma interrupt**, donde indicamos que las instrucciones escritas a continuación comenzarán en el vector de interrupciones.

TIPO DE PUNTERO	EJEMPLO	TAMAÑO
Puntero a memoria de datos	Char * ptr;	16 bits
Puntero a memoria de programa cercano	rom near char * n_rom_ptr;	16 bits
Puntero a memoria de programa lejano	rom far char * f_rom_ptr;	24 bits

TABLA 2. **Ejemplos de distintos tipos de punteros con respecto a la memoria de datos y de programa, y las longitudes en bits de cada uno de ellos.**

Primer **programa en C**

Aprenderemos el entorno de programación en C a través de un ejemplo práctico. Luego, realizaremos una simulación con el **MPLAB SIM**.

INTRODUCCIÓN AL EJERCICIO

Para nuestro primer programa en C, vamos a realizar un contador decimal, que posee dos pulsadores para incrementar o disminuir la cuenta. El estado de la cuenta será visualizado por medio de un **display de 7 segmentos**.

En el esquema del ejercicio (**Figura 4**) podemos observar los pulsadores, uno de los cuales tendrá la función de incrementar la cuenta, y el otro, de disminuirla. El **display de 7 segmentos** estará conectado al puerto **B** del **PIC18LF4620**.

DESCRIPCIÓN DEL CÓDIGO

En la **Figura 14** se muestra el diagrama de flujo del ejercicio, a partir de cuyo código nos guiaremos para resolver el problema. El código comienza configurando los puertos del PIC; las entradas de los pulsadores serán **RA1** y **RA2,** y la salida del programa será el **puerto B**.

Luego, entramos en un **bucle infinito,** donde actualizamos el valor de los **latchs** del puerto **B** con el valor actual de la cuenta en formato de **7 segmentos**. A continuación, nos quedamos esperando el cambio en lo pulsadores. Basta con oprimir uno de ellos para que la condición sea verdadera, y el programa pase a interrogar cuál es el que se presionó. Una vez identificado, se realiza la función correspondiente de incrementar o disminuir el estado de la cuenta.

Si la cuenta se excede de su valor máximo, inicializamos el contador; pero si es menor que **0** (cero), retornamos con el valor **9**.

Cuando pasamos al código del ejercicio, creamos una tabla con la conversión de la cuenta a un display de 7 segmentos, que declaramos en la memoria de programa. Una variable **Índice** recorre las posiciones de la tabla para devolvernos su valor en código de 7 segmentos; la utilizaremos como el contador de nuestro programa.





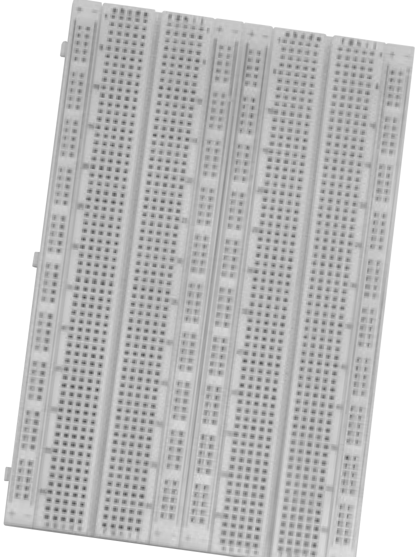
DIRECTIVA #PRAGMA CONFIG

Los bits de configuración se utilizan para configurar muchas funcionalidades del PIC; por ejemplo, podemos establecer el tipo de oscilador, activar protecciones a la memoria de programa y realizar muchas otras funciones que poseen los PIC18F.

FIGURA 14. Este es el esquema que debemos tomar como referencia para el desarrollo de la práctica.

Para detectar el cambio de los pulsadores, realizamos un **AND lógico** entre los estados **RA1** y **RA2** dentro de la expresión del **while.** Entonces, basta con oprimir uno de los pulsadores para que la expresión entre paréntesis sea falsa, y el programa salte a la próxima instrucción.

Las entradas de los pulsadores serán RA1 y RA2, y la salida del programa será el puerto B



CÓDIGO RESULTANTE

Vamos a mostrar el código resultante del ejercicio, que, luego, podemos simular en **MPLAB SIM** para verificar su funcionamiento. Es importante comentar todas las líneas de código para no olvidar cuál es la función de cada sentencia. Observemos también la posición de memoria que ocupa la tabla: la colocamos bien alejada del código de programa para que no se superpongan. Siempre debemos controlar estas posiciones de memoria para no tener problemas.

```
/* Includes */
#include “p18f4620.h” // Incluimos archivo con
los registros del PIC

/* Declaraciones */
// declaramos datos constante en memoria
de programa comenzando en la
// direccion 0x180
#pragma romdata Tabla_Display = 0x180
const rom unsigned char Tabla[10] = {
0b00111111, //0

        0b00000110, //1
        0b01011011, //2
        0b01001111, //3
        0b01100110, //4
        0b01101101, //5
        0b01111101, //6
        0b00000111, //7
        0b01111111, //8
        0b01100111};//9

/* Programa principal */
#pragma code  // zona de código
```

```
void main (void)
{
char Indice = 0; // variable de 8 bits

 TRISB = 0b00000000; // PORTB como salida

ADCON1=0x0F;        //Configuramos
todos los pines como digitales

TRISA = 0b00000110;
// RA1,RA2 entradas

        while (1)
        {// Entramos en un bucle infinito

        // usamos la tabla para
        obtener las salidas en los LEDs y
        // la recorremos con
        la variable Indice
        LATB = Tabla[Indice];
        while (PORTAbits.RA1 != 1 &&
        PORTAbits.RA2 != 1);
        // No hacemos nada hasta
        que se produce un cambio en
        // cualquiera de los pulsadores

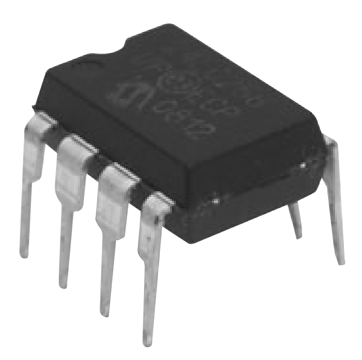
        if (PORTAbits.RA1 == 1)
        // Se pulso RA1?
        {
                Indice++;
                // incrementamos el indice
                if (Indice == 10)
                Indice = 0; //
                inicializamos Indice
        }
```

```
        if (PORTAbits.RA2 == 1)
        // Se pulso RA2?
        {
                Indice—;
                // Decrementamos el indice
                if (Indice < 0)
                Indice = 9;
                // para la cuenta regresiva
        }
    }
}
```

Con este primer ejercicio, aprendimos cuál es la estructura de un programa en lenguaje de alto nivel para los microcontroladores, en el que las sentencias son fáciles de seguir. El próximo paso es compilar el proyecto y realizar la simulación (**Figura 15**).

SIMULACIÓN DE PROGRAMAS EN MPLAB SIM

Vamos a utilizar el **MPLAB SIM** para simular nuestro programa. Pero antes debemos compilar el programa, por lo que debemos crear un nuevo proyecto en **MPLAB**.



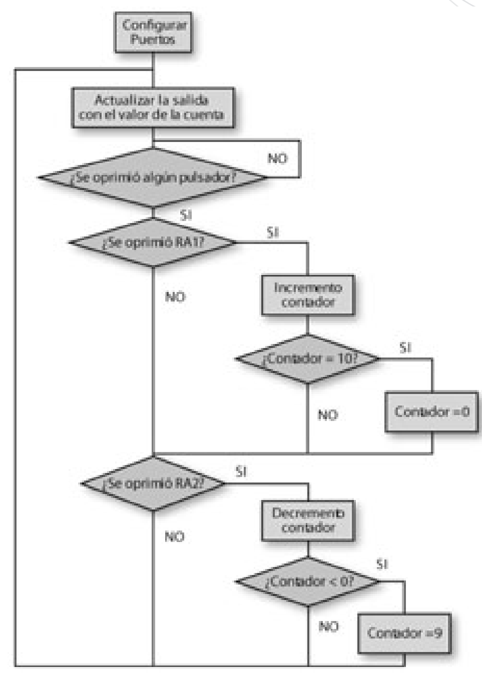


FIGURA 15. **En este diagrama podemos seguir el flujo del ejercicio que estamos realizando.**

Comenzamos seleccionando **`Project/Project Wizard`** en el entorno de **MPLAB**. Luego, elegimos el PIC con el que vamos a trabajar; en este caso, es el **PIC18F4620**. En el siguiente paso, debemos escoger el lenguaje de programación, para lo cual nos dirigimos a **`Microchip C18 Toolsuite`**.

Indicamos el nombre del nuevo proyecto y la ubicación que tendrá en la PC. Cuando el programa nos pida agregar un archivo, colocamos el código fuente que acabamos de crear y, por último, presionamos **`Finalizar`**.

Solo resta un paso más, que es poner el archivo **linker** necesario para que el **C18** pueda compilar. Para hacerlo, vamos a la ventana `Project` y, en la carpeta **Linker Script**, hacemos clic con el botón secundario del mouse y elegimos `Add Files`. Buscamos el archivo **18f4620i.lkr** en la carpeta **LKR**, dentro del directorio **MCC18**.

Una vez hecho todo esto, compilamos el proyecto, seleccionando `Build All`. Para comenzar con la simulación, vamos a `Debugger/Select tool` y elegimos `MPLAB SIM`. Veremos que aparecen nuevos iconos en la barra de herramientas, que sirven para controlar el proceso de simulación. Para evitar pasar por todos los archivos de arranque, podemos colocar un **breakpoint** en la primera instrucción de la función **main** y correr el programa con la tecla **run** hasta detenerse allí.

Special Function Registers

Address	SFR Name	Hex
	TMR3_Prescale	0x00
F80	PORTA	0x09
F81	PORTB	0x3F
F82	PORTC	0x00
F83	PORTD	0x00
F84	PORTE	0x00
F89	LATA	0x09
F8A	LATB	0x3F
F8B	LATC	0x00
F8C	LATD	0x00
F8D	LATE	0x00
F92	TRISA	0x06
F93	TRISB	0x00
F94	TRISC	0xFF
F95	TRISD	0xFF
F96	TRISE	0x07
F9B	OSCTUNE	0x00
F9D	PIE1	0x00
F9E	PIR1	0x00
F9F	IPR1	0xFF
FA0	PIE2	0x00
FA1	PIR2	0x00
FA2	IPR2	0xDF
FA6	EECON1	0x00
FA7	EECON2	0x00
FA8	EEDATA	0x00
FA9	EEADR	0x00
FAA	EEADRH	0x00
FAB	RCSTA	0x00

FIGURA 16. En MPLAB, las modificaciones de los registros se visualizan en la ventana de `SFR`**. Para ver los registros más importantes, configuramos la ventana** `Watch`**.**

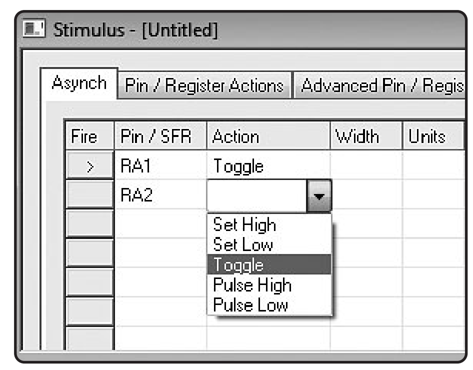


FIGURA 17. Desde la ventana `Stimulus` **podemos simular cambios asincrónicos en los pines del PIC. Esto es muy útil cuando simulamos entradas digitales. En la imagen vemos cómo configurar la entrada de los pulsadores en el programa.**

Es posible simular los pulsadores en la ventana `Debugger/Stimulus`, seleccionando `New Workbook`. En la solapa `Asynch` buscamos la columna `Pin/SFR` y elegimos, por ejemplo, el pin **RA1**, mientras que en `Action` elegimos la acción que realizará el pulsador, como poner un nivel alto (**Figura 16**). Cada vez que oprimamos `Fire`, durante la simulación, se llevará a cabo la acción seleccionada (**Figura 17**).

PROGRAMADOR MCE PDX USB

En el mercado es posible encontrar muchas herramientas de desarrollo para los microcontroladores

El programador MCE PDX se conecta a la PC a través del puerto USB y no necesita alimentación externa

PIC. Los programadores comerciales más populares de Microchip son actualmente **MPLAB ICD2** y **PicKit 2**, que no solo graban el PIC, sino que también permiten realizar la depuración de código en circuito (*debugger in-circuit*).

Una de las ventajas de poder depurar código en el circuito es detectar esos errores que son ajenos al programa. Si bien estos programadores resultan muy útiles, su costo es elevado, pero en el mercado podemos encontrar herramientas con las mismas funciones a un costo mucho menor. Este es el caso del **MCE PDX USB**, de la empresa **MC Electronics**. Esta versión comercial es una buena alternativa a las de Microchip, porque además de poseer la funcionalidad de depurar código en circuito, también agrega funciones alternativas para detectar errores. Puede usarse, además, como un simple analizador de estados lógicos de tres canales. Soporta una gran cantidad de microcontroladores PIC y es totalmente compatible con el entorno de programación MPLAB.

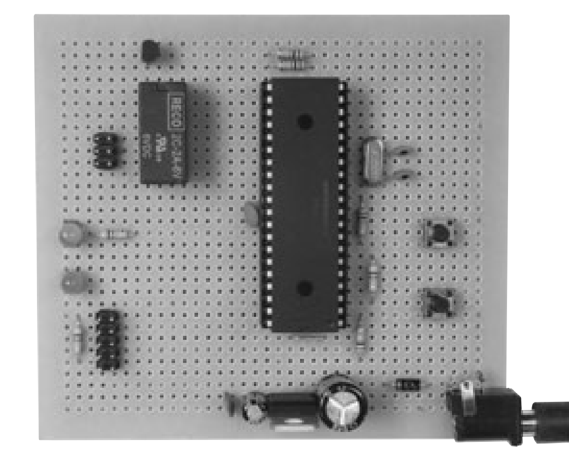
El programador MCE PDX se conecta a la PC a través del puerto USB y no necesita alimentación externa, ya que la toma directamente del puerto. También posee un regulador de tensión interno de 2,5 volts a 5 volts, lo que permite grabar microcontroladores que se alimenten a 3,3 volts, muy comunes en estos días.

Entonces, una herramienta como esta nos será de gran ayuda en nuestro taller si queremos profundizar en el conocimiento de los microcontroladores PIC.

DEPURACIÓN EN CIRCUITO

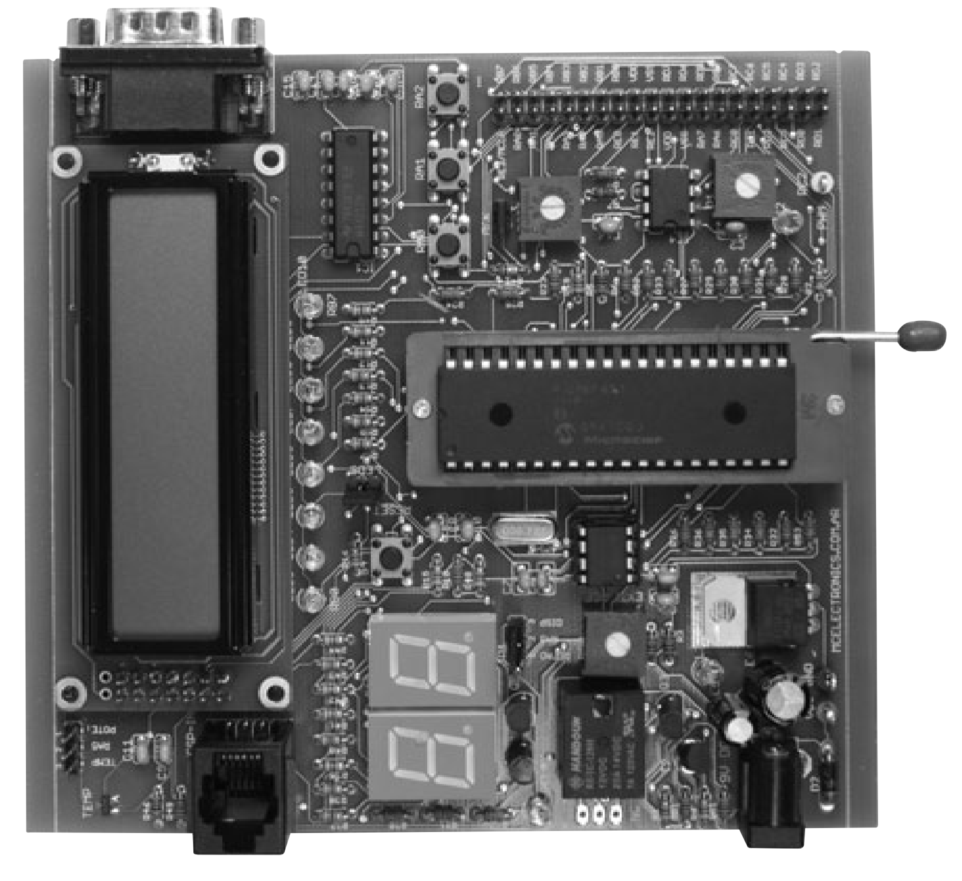
Muchos de los microcontroladores PIC incorporan **ICSP** (*In Circuit Serial Programming*, programación serie en circuito) o **LVP** (*Low Voltage Programming*, programación a bajo voltaje). Estas características permiten programar el PIC en el circuito, sin necesidad de retirarlo. Solo hay que agregar en el **PCB** un conector con las cinco líneas para la programación del PIC.

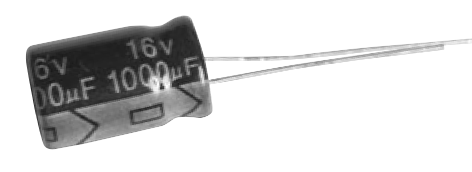
En la programación **ICSP**, utilizamos los pines **RB6** y **RB7** como reloj y datos, y el pin **MCLR** para activar el modo programación, por medio de una tensión de entre 12 y 13 volts.



Pero esta comunicación serie que tenemos con el PIC no solo nos permite programarlo, sino que también nos da la posibilidad de efectuar una depuración del código en circuito (*In-Circuit Debugging*). La herramienta para hacerlo es el programador **MCE PDX USB**, que ya tiene disponibles conectores externos para la programación y depuración de código en circuito.

ICSP y LVP son características que permiten programar el PIC en el circuito, sin necesidad de retirarlo





Lo importante es que este programador es compatible con la interfaz del **MPLAB**, y las herramientas de depuración son las mismas que estudiamos con el simulador **MPLAB SIM**.

DEPURACIÓN IN-CIRCUIT DEL PROGRAMA EN C

Para realizar la depuración del código, debemos contar con un programador que nos permita hacer este proceso en circuito, como el **MCE PDX,** y conectarlo a nuestra placa experimental a través de las cinco líneas de programación.

En este caso, utilizamos una placa comercial que posee pulsadores y display de 7 segmentos para efectuar la depuración del código. Una vez que tenemos conectados el programador y la placa experimental, nos resta enchufar el cable USB del programador a la PC.

Ejecutamos el **MPLAB** y abrimos nuestro primer proyecto en C, el contador ascendente/descendente. Luego, elegimos la herramienta de depuración en el **MPLAB, para lo cual** nos dirigimos al menú **`Debugger/Select Tool`** y elegimos **`7 PicKit 2`**. En la ventana **`Output`** del **MPLAB**, debemos visualizar el estado de la conexión del programador y verificar que no se produjeron errores.

Observaremos que aparecen las mismas herramientas para la depuración de código que teníamos con **MPLAB SIM** y que, además, se agregaron otras (propias del programador).

Antes de comenzar, debemos compilar el código haciendo clic en **`Project/Build All`**. Luego, programamos el micro desde **`Debugger/Program`**, para que el micro se grabe con el código y nos permita efectuar la depuración.

Ahora podemos comenzar con la simulación del código. es importante destacar que este microcontrolador nos permite poner hasta **3 breakpoints** o puntos de detención de programa.

Si corremos la depuración con **`run`**, veremos cómo se ejecuta el programa en la placa y podremos detenerlo en cualquier instante, oprimiendo el icono **`halt`**.

También tenemos la opción de ejecutar el código línea por línea y ver cómo se modifican los registros de funciones especiales (SFR). La ventana para visualizar los SFR se selecciona desde **`View/Special Functions Registers`**.

En el momento de llegar al **while,** que espera el cambio de los estados de los pulsadores, podemos oprimir cualquiera de ellos en la placa y ver de qué manera se comporta el código ante ese estímulo que le imponemos.

Veremos que esta forma de buscar errores en el código es más efectiva, porque nos permite observar el comportamiento del PIC con los periféricos externos, y encontrar si algún error es producto del código o del hardware.

Cuando estemos en el modo depuración, notaremos que los pines **RB6** y **RB7** se utilizan para la comunicación, por lo que no podremos simularlos como entradas o salidas digitales.

Multiple **choice**

1 **¿Cuántos pines posee el encapsulado del microcontrolador PIC18F4620?**

a- 10.
b- 20.
c- 30.
d- 40.

2 **¿Cuántos modos de oscilador tiene el microcontrolador PIC18F4620?**

a- 10.
b- 20.
c- 30.
d- 40.

3 **¿Cuántas señales diferentes genera el oscilador interno del microcontrolador PIC18F4620?**

a- 1.
b- 2.
c- 3.
d- 5.

4 **¿Cuántos puertos de entrada y salida posee el microcontrolador PIC18F4620?**

a- 1.
b- 2.
c- 3.
d- 5.

5 **¿Cuántos tipos de memoria tiene el microcontrolador PIC18F4620?**

a- 1.
b- 2.
c- 3.
d- 5.

6 **¿Cuál de las siguientes no es una memoria del microcontrolador PIC18F4620?**

a- Memoria de programas.
b- RAM de datos.
c- EEPROM de datos.
d- ROM de datos.

Respuestas: 1 d, 2 a, 3 b, 4 d, 5 c, 6 d.

Servicios **al lector**



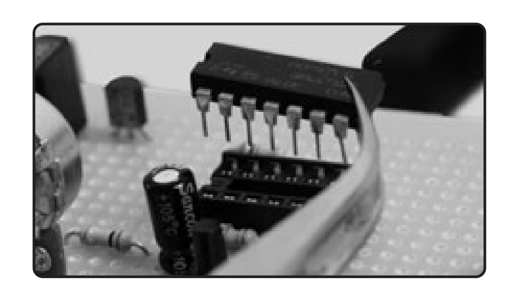
Encontraremos información adicional relacionada con el contenido, que servirá para complementar lo aprendido.

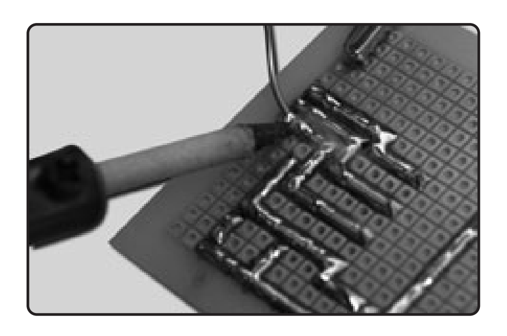
Índice **temático**

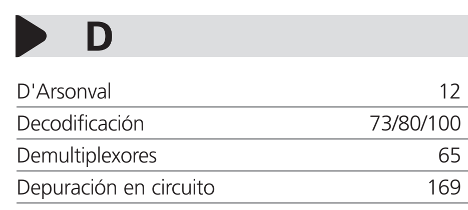
A

B

C





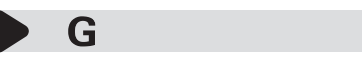


D

E

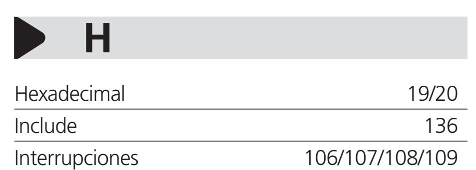
F



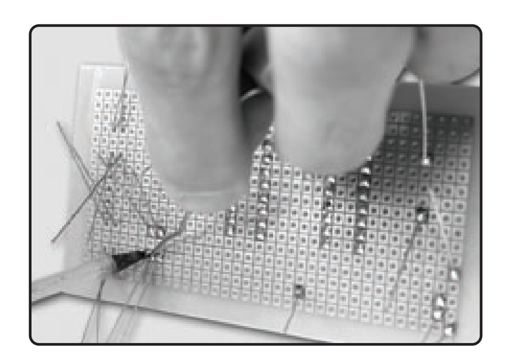


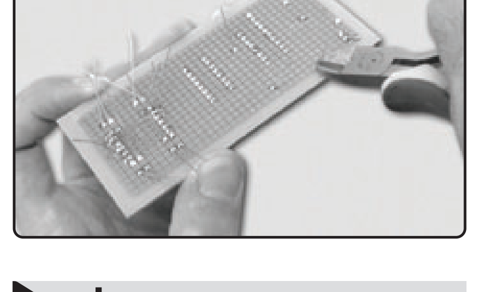
G





H



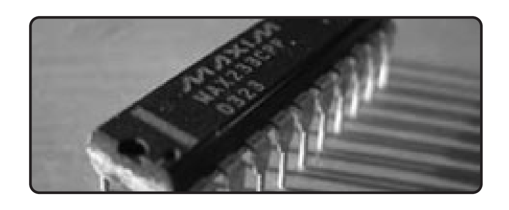




J

L

M


MAXIM

Sitios web **recomendados**



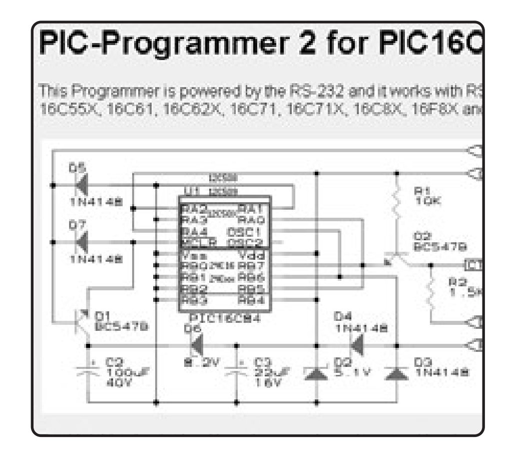
MICROCHIP
www.microchip.com



Sitio oficial del fabricante de los microcontroladores PIC. En él podemos encontrar mucha información, hojas de datos, notas de aplicación, herramientas de desarrollo, etcétera.

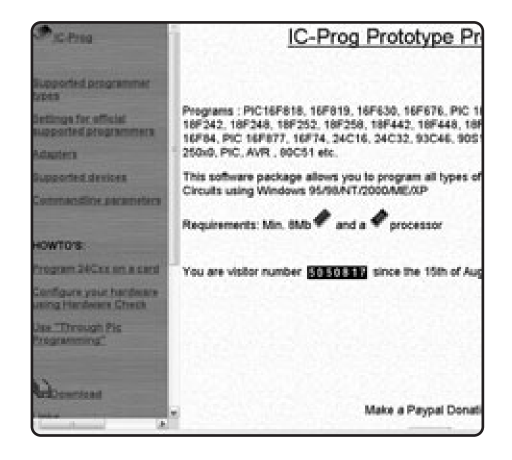


JDM
www.jdm.homepage.dk/newpic.htm



Sitio del creador de los grabadores JDM para PIC. Aquí podemos encontrar circuitos de los grabadores y otra información de interés.

IC-PROG
www.ic-prog.com/index1.htm



Sitio oficial del programa IC-Prog. Desde aquí podemos descargar el programa y encontrar información de los programadores soportados, archivos de ayuda, etcétera.

FOROS DE ELECTRÓNICA
www.forosdeelectronica.com



Foros de preguntas y respuestas relacionadas con cualquier tema de electrónica y, por supuesto, con microcontroladores. También hay tutoriales, manuales y proyectos.



DATA SHEET CATALOG

www.datasheetcatalog.com



Desde esta página podemos descargar hojas de datos de circuitos integrados, transistores, diodos y, en general, de cualquier componente electrónico de forma gratuita.



DATASHEET 4 YOU

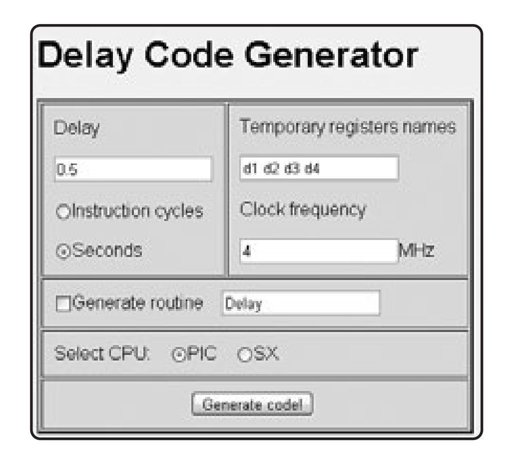
www.datasheet4u.com



Otro sitio en el que podemos encontrar una extensa colección de hojas de datos de componentes electrónicos, las cuales tenemos la posibilidad de descargar en forma gratuita.



DELAY CODE GENERATOR
www.golovchenko.org/cgi-bin/delay



Generador de subrutinas de retardo para microcontroladores PIC en línea.



LABCENTER ELECTRONICS
www.labcenter.co.uk



Página oficial del programa de simulación de circuitos electrónicos y microcontroladores Proteus. Desde aquí podemos descargar una versión demo y encontrar información del programa.



IEEPROTEUS
www.ieeproteus.com



Sitio totalmente en español dedicado al programa de simulación y al diseño de circuitos electrónicos Proteus.



UCONTROL
www.ucontrol.com.ar



Sitio dedicado al tema de la electrónica y los microcontroladores. En él podemos encontrar proyectos, tutoriales y mucho más.

TODOPIC FOROS

www.todopic.com.ar/foros



Foros de preguntas y respuestas dedicados principalmente a temas relacionados con los microcontroladores PIC.



X-ROBOTICS

www.x-robotics.com



Sitio dedicado a la robótica y a los microcontroladores PIC. En él podemos encontrar proyectos, ejemplos, tutoriales y más.

CLAVES PARA COMPRAR
UN LIBRO DE COMPUTACIÓN



1 SOBRE EL AUTOR Y LA EDITORIAL

Revise que haya un cuadro "sobre el autor", en el que se informe sobre su experiencia en el tema. En cuanto a la editorial, es conveniente que sea especializada en computación.



2 PRESTE ATENCIÓN AL DISEÑO

Compruebe que el libro tenga guías visuales, explicaciones paso a paso, recuadros con información adicional y gran cantidad de pantallas. Su lectura será más ágil y atractiva que la de un libro de puro texto.



3 COMPARE PRECIOS

Suele haber grandes diferencias de precio entre libros del mismo tema; si no tiene el valor en tapa, pregunte y compare.



4 ¿TIENE VALORES AGREGADOS?

Desde un sitio exclusivo en la Red, un Servicio de Atención al Lector, la posibilidad de leer el sumario en la Web para evaluar con tranquilidad la compra, y hasta la presencia de adecuados índices temáticos, todo suma al valor de un buen libro.



5 VERIFIQUE EL IDIOMA

No solo el del texto; también revise que las pantallas incluidas en el libro estén en el mismo idioma del programa que usted utiliza.



usershop.redusers.com

VISITE NUESTRO SITIO WEB

- Vea información más detallada sobre cada libro de este catálogo.
- Obtenga un capítulo gratuito para evaluar la posible compra de un ejemplar.
- Conozca qué opinaron otros lectores.
- Compre los libros sin moverse de su casa y con importantes descuentos.
- Publique su comentario sobre el libro que leyó.
- Manténgase informado acerca de las últimas novedades y los próximos lanzamientos.

TAMBIÉN PUEDE CONSEGUIR NUESTROS LIBROS EN KIOSCOS O PUESTOS DE PERIÓDICOS, LIBRERÍAS, CADENAS COMERCIALES, SUPERMERCADOS Y CASAS DE COMPUTACIÓN.



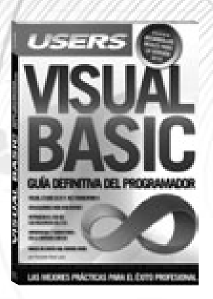
LLEGAMOS A TODO EL MUNDO VÍA OCA * Y  **

* SOLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

usershop.redusers.com





Visual Basic

Este libro está escrito para aquellos usuarios que quieran aprender a programar en VB.NET. Desde el IDE de programación hasta el desarrollo de aplicaciones del mundo real en la versión 2010 de Visual Studio, todo está contemplado para conocer en profundidad VB.NET al finalizar la lectura.

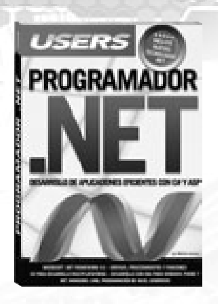
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-57-2



Microcontroladores

Este manual es ideal para aquellos que quieran iniciarse en la programación de microcontroladores. A través de esta obra, podrán conocer los fundamentos de los sistemas digitales, aprender sobre los microcontroladores PIC 16F y 18F, hasta llegar a conectar los dispositivos de forma inalámbrica, entre muchos otros proyectos.

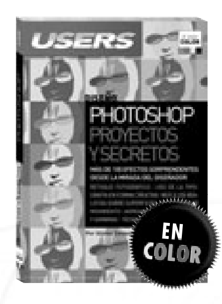
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-56-5



Programador.NET

Este libro está dirigido a todos aquellos que quieran iniciarse en el desarrollo bajo lenguajes Microsoft. A través de los capítulos del manual, aprenderemos sobre POO y la programación con tecnologías .NET, su aplicación, cómo interactúan entre sí y de qué manera se desenvuelven con otras tecnologías existentes.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-26-8



Photoshop: proyectos y secretos

En esta obra aprenderemos a utilizar Photoshop, desde la original mirada de la autora. Con el foco puesto en la comunicación visual, a lo largo del libro adquiriremos conocimientos teóricos, al mismo tiempo que avanzaremos sobre la práctica, con todos los efectos y herramientas que ofrece el programa.

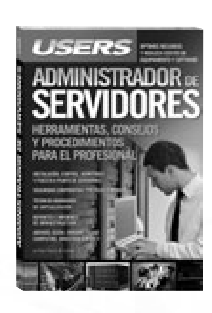
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-25-1



WordPress

Este manual está dirigido a todos aquellos que quieran presentar sus contenidos o los de sus clientes a través de WordPress. En sus páginas el autor nos enseñará desde cómo llevar adelante la administración del blog hasta las posibilidades de interacción con las redes sociales.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-18-3



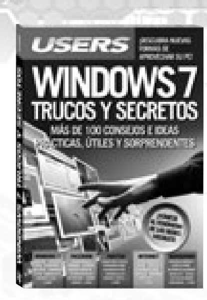
Administrador de servidores

Este libro es la puerta de acceso para ingresar en el apasionante mundo de los servidores. Aprenderemos desde los primeros pasos sobre la instalación, configuración, seguridad y virtualización; todo para cumplir el objetivo final de tener el control de los servidores en la palma de nuestras manos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-19-0

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Windows 7: Trucos y secretos

Este libro está dirigido a todos aquellos que quieran sacar el máximo provecho de Windows 7, las redes sociales y los dispositivos ultraportátiles del momento. A lo largo de sus páginas, el lector podrá adentrarse en estas tecnologías mediante trucos inéditos y consejos asombrosos.

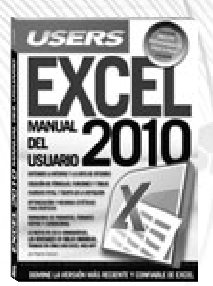
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-17-6



Desarrollo PHP + MySQL

Este libro presenta la fusión de dos de las herramientas más populares para el desarrollo de aplicaciones web de la actualidad: PHP y MySQL. En sus páginas, el autor nos enseñará las funciones del lenguaje, de modo de tener un acercamiento progresivo, y aplicar lo aprendido en nuestros propios desarrollos.

→ COLECCIÓN: MANUALES USERS
→ 432 páginas / ISBN 978-987-1773-16-9



Excel 2010

Este manual resulta ideal para quienes se inician en el uso de Excel, así como también para los usuarios que quieran conocer las nuevas herramientas que ofrece la versión 2010. La autora nos enseñará desde cómo ingresar y proteger datos hasta la forma de imprimir ahorrando papel y tiempo.

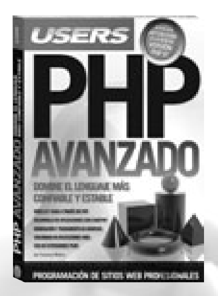
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-15-2



Técnico Hardware

Esta obra es fundamental para ganar autonomía al momento de reparar la PC. Aprenderemos a diagnósticar y solucionar las fallas, así como a prevenirlas a través del mantenimiento adecuado, todo explicado en un lenguaje práctico y sencillo.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-14-5



PHP Avanzado

Este libro brinda todas las herramientas necesarias para acercar al trabajo diario del desarrollador los avances más importantes incorporados en PHP 6. En sus páginas, repasaremos todas las técnicas actuales para potenciar el desarrollo de sitios web.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-1773-07-7



AutoCAD

Este manual nos presenta un recorrido exhaustivo por el programa más difundido en dibujo asistido por computadora a nivel mundial, en su versión 2010. En sus páginas, aprenderemos desde cómo trabajar con dibujos predeterminados hasta la realización de objetos 3D.

→ COLECCIÓN: MANUALES USERS
→ 384 páginas / ISBN 978-987-1773-06-0

usershop@redusers.com





Windows Seven Avanzado

Esta obra nos presenta un recorrido exhaustivo que nos permitirá acceder a un nuevo nivel de complejidad en el uso de Windows 7. Todas las herramientas son desarrolladas con el objetivo de acompañar al lector en el camino para ser un usuario experto.

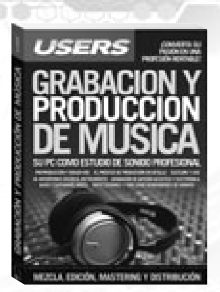
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-08-4



Photoshop

En este libro aprenderemos sobre las más novedosas técnicas de edición de imágenes en Photoshop. El autor nos presenta de manera clara y práctica todos los conceptos necesarios, desde la captura digital hasta las más avanzadas técnicas de retoque.

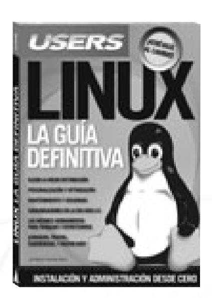
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-05-3



Grabación y producción de música

En este libro repasaremos todos los aspectos del complejo mundo de la producción musical. Desde las cuestiones para tener en cuenta al momento de la composición, hasta la mezcla y el masterizado, así como la distribución final del producto.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-04-6



Linux

Este libro es una completa guía para migrar e iniciarse en el fascinante mundo del software libre. En su interior, el lector conocerá las características de Linux, desde su instalación hasta las opciones de entretenimiento, con todas las ventajas de seguridad que ofrece el sistema.

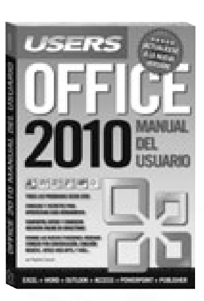
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-8-6



Premiere + After Effects

Esta obra nos presenta un recorrido detallado por las aplicaciones audiovisuales de Adobe: Premiere Pro, After Effects y Soundbooth. Todas las técnicas de los profesionales, desde la captura de video hasta la creación de efectos, explicadas de forma teórica y práctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-9-3



Office 2010

En este libro aprenderemos a utilizar todas las aplicaciones de la suite, en su versión 2010. Además, su autora nos mostrará las novedades más importantes, desde los minigráficos de Excel hasta Office Web Apps, todo presentado en un libro único.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-26013-6-2

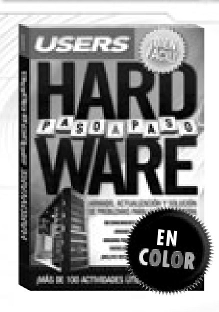




200 Respuestas: Blogs

Esta obra es una completa guía que responde a las preguntas más frecuentes de la gente sobre la forma de publicación más poderosa de la Web 2.0. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

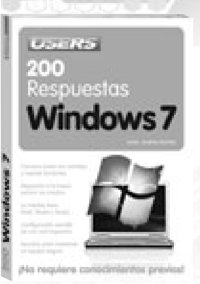
→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-663-037-5



Hardware paso a paso

En este libro encontraremos una increíble selección de actividades que abarcan todos los aspectos del hardware. Desde la actualización de la PC hasta el overclocking de sus componentes, todo en una presentación nunca antes vista, realizada íntegramente con procedimientos paso a paso.

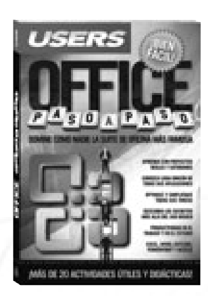
→ COLECCIÓN: PASO A PASO
→ 320 páginas / ISBN 978-987-663-034-4



200 Respuestas: Windows 7

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos conocer para dominar la última versión del sistema operativo de Microsoft. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-663-035-1



Office paso a paso

Este libro presenta una increíble colección de proyectos basados en la suite de oficina más usada en el mundo. Todas las actividades son desarrolladas con procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: PASO A PASO
→ 320 páginas / ISBN 978-987-663-030-6



101 Secretos de Hardware

Esta obra es la mejor guía visual y práctica sobre hardware del momento. En su interior encontraremos los consejos de los expertos sobre las nuevas tecnologías, las soluciones a los problemas más frecuentes, cómo hacer overclocking, modding, y muchos más trucos y secretos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-029-0



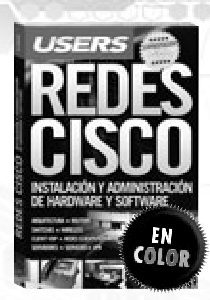
Access

Este manual nos introduce de lleno en el mundo de Access para aprender a crear y administrar bases de datos de forma profesional. Todos los secretos de una de las principales aplicaciones de Office, explicados de forma didáctica y sencilla.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-025-2

¡Léalo antes Gratis!

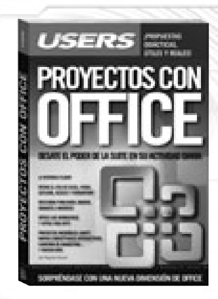
En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Redes Cisco

Este libro permitirá al lector adquirir todos los conocimientos necesarios para planificar, instalar y administrar redes de computadoras. Todas las tecnologías y servicios Cisco, desarrollados de manera visual y práctica en una obra única.

→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-663-024-5



Proyectos con Office

Esta obra nos enseña a usar las principales herramientas de Office a través de proyectos didácticos y útiles. En cada capítulo encontraremos la mejor manera de llevar adelante todas las actividades del hogar, la escuela y el trabajo.

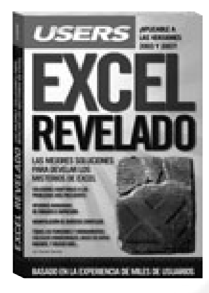
→ COLECCIÓN: MANUALES USERS
→ **352** páginas / ISBN 978-987-663-023-8



Dreamweaver y Fireworks

Esta obra nos presenta las dos herramientas más poderosas para la creación de sitios web profesionales de la actualidad. A través de procedimientos paso a paso, nos muestra cómo armar un sitio real con Dreamweaver y Fireworks sin necesidad de conocimientos previos.

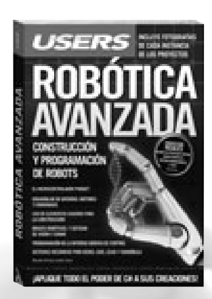
→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-663-022-1



Excel revelado

Este manual contiene una selección de más de 150 consultas de usuarios de Excel y todas las respuestas de Claudio Sánchez, un reconocido experto en la famosa planilla de cálculo. Todos los problemas encuentran su solución en esta obra imperdible.

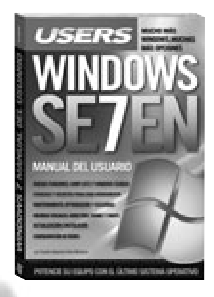
→ COLECCIÓN: MANUALES USERS
→ **336** páginas / ISBN 978-987-663-021-4



Robótica avanzada

Esta obra nos permitirá ingresar al fascinante mundo de la robótica. Desde el ensamblaje de las partes hasta su puesta en marcha, todo el proceso está expuesto de forma didáctica y sencilla para así crear nuestros propios robots avanzados.

→ COLECCIÓN: MANUALES USERS
→ **352** páginas / ISBN 978-987-663-020-7



Windows 7

En este libro encontraremos las claves y los secretos destinados a optimizar el uso de nuestra PC tanto en el trabajo como en el hogar. Aprenderemos a llevar adelante una instalación exitosa y a utilizar todas las nuevas herramientas que incluye esta versión.

→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-663-015-3

usershop@redusers.com



usershop.redusers.com

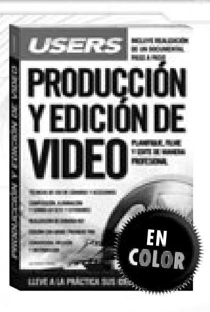




De Windows a Linux

Esta obra nos introduce en el apasionante mundo del software libre a través de una completa guía de migración, que parte desde el sistema operativo más conocido: Windows. Aprenderemos cómo realizar gratuitamente aquellas tareas que antes hacíamos con software pago.

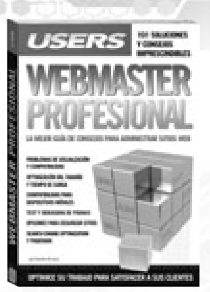
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-013-9



Producción y edición de video

Un libro ideal para quienes deseen realizar producciones audiovisuales con bajo presupuesto. Tanto estudiantes como profesionales encontrarán cómo adquirir las habilidades necesarias para obtener una salida laboral con una creciente demanda en el mercado.

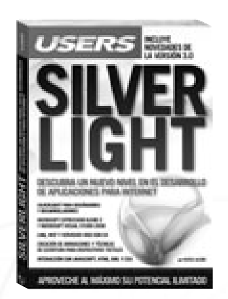
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-012-2



Webmaster Profesional

Esta obra explica cómo superar los problemas más frecuentes y complejos que enfrenta todo administrador de sitios web. Ideal para quienes necesiten conocer las tendencias actuales y las tecnologías en desarrollo que son materia obligada para dominar la Web 2.0.

→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-011-5



Silverlight

Este manual nos introduce en un nuevo nivel en el desarrollo de aplicaciones interactivas a través de Silverlight, la opción multiplataforma de Microsoft. Quien consiga dominarlo creará aplicaciones visualmente impresionantes, acordes a los tiempos de la incipiente Web 3.0.

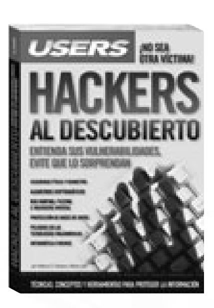
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-010-8



Flash Extremo

Este libro nos permitirá aprender a fondo Flash CS4 y ActionScript 3.0 para crear aplicaciones web y de escritorio. Una obra imperdible sobre uno de los recursos más empleados en la industria multimedia que nos permitirá estar a la vanguardia del desarrollo.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-009-2



Hackers al descubierto

Esta obra presenta un panorama de las principales técnicas y herramientas utilizadas por los hackers, y de los conceptos necesarios para entender su manera de pensar, prevenir sus ataques y estar preparados ante las amenazas más frecuentes.

→ COLECCIÓN: MANUALES USERS
Este manual es una pieza imprescindi-
→ 352 páginas / ISBN 978-987-663-008-5

MICROCONTROLADORES

En esta obra veremos las diferencias entre los sistemas analógicos y los digitales. Además, analizaremos los conceptos en los que se basa la electrónica digital, ingresaremos en el mundo de las memorias y aprenderemos a programar el microcontrolador PIC16F, uno de los más populares del mercado.

DENTRO DEL LIBRO ENCONTRARÁ

■ Sistemas analógicos y binarios ■ Compuertas lógicas ■ Temporizador con 4093 ■ Aritmética binaria ■ Circuitos secuenciales ■ Almacenamiento digital ■ Aplicación de memorias PROM ■ Memorias FRAM ■ Microcontroladores ■ Unidades de entrada-salida ■ Interrupciones ■ Microcontroladores PIC 16F ■ Simulador de hogar a leña ■ MPLAB ■ Grabador de PICs ■ PIC18F



ADEMÁS

ELECTRÓNICA PRÁCTICA
Aprenda a analizar, simular y construir circuitos

PROYECTOS CON MICROCONTROLADORES
Aprenda a desarrollar sus propias aplicaciones

NETWORKING CON MICROCONTROLADORES
Descubra cómo acceder remotamente a sus equipos

SOBRE LA COLECCIÓN: ELECTRÓNICA

■ **Aprendizaje guiado mediante explicaciones claras y concisas** ■ **Proyectos prácticos basados en necesidades reales** ■ **Consejos de los profesionales** ■ **Infografías y procedimientos paso a paso** ■ **Producciones fotográficas profesionales**

MICROCONTROLLERS



In this book we will learn the basics of microcontrollers as well as the practical aspects to bear in mind when programming and using them in real projects. Furthermore, we will be trained on how to use PIC16F with Assembler and PIC18F with C.





ISBN 978-987-1773-22-0